ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

APLICACIÓN IVR (INTERACTIVE VOICE RESPONSE) PARA LA CONSULTA TELEFÓNICA DE NOTAS CON SÍNTESIS TEXTO-VOZ EN TIEMPO REAL

AUTORA: Eva Mª Navarro Albaladejo
DIRECTOR: Pablo Pavón Mariño
CODIRECTOR: José Juan Sánchez Manzanares

Febrero / 2006

Autor	Eva Mª Navarro Albaladejo
E-mail del Autor	ivenmeri@gmail.com
Director(es)	Pablo Pavón Mariño
E-mail del Director	Pablo.Pavón@upct.com
Codirector(es)	José Juan Sánchez Manzanares
Título del PFC	Aplicación IVR (Interactive Voice Response) para la consulta telefónica de notas con síntesis texto-voz en tiempo real
Descriptores	CTI, IVR, Dialogic, VerbioTTS, ODBC

Resúmen

En este proyecto fin de carrera se ha mejorado y actualizado un sistema IVR que permite la consulta telefónica de notas, realizada anteriormente por otro alumno de esta misma universidad.

Para el acceso a los datos se ha utilizado una conexión ODBC, que nos permitará acceder a los datos independientemente de cual sea el motor de bases de datos utilizados - en esta versión el sistema gestor de bases de datos es Microsoft Access. Hemos usado la tarjeta de Dialogic D4/PCI, que es una de las tarjetas más recomendadas del mercado para aplicaciones IVR; y un sintetizador de texto a voz en tiempo real VerbioTTS de la casa Verbio.

La aplicación se ha desarrollado en lenguaje C y en modo de consola con programación estructurada no orientada a objetos.

Titulación	Ingeniero Técnico de Telecomunicación, Esp. Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Febrero- 2006

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Capítulo 1: Sistemas CTI-IVR y VerbioTTS

1.1 Introducción

En épocas de exámenes las clases se suspenden y los alumnos normalmente sólo acuden a la universidad para la realización de estos y cuando ya se han realizado para la consulta de notas e información de la fechas de revisión. Hoy en día los alumnas pueden consultar sus notas desde Red Campus, desde la página del profesor ó incluso desde la herramienta “Aula Virtual” en la que no se encuentran registradas todas las asignatura pero si muchas de ellas.

Lo que se va aproponer en este proyecto fin de carrera es una alternativa más para la consulta de notas sin tener que desplazarse a la universidad. El acceso al teléfono, hoy día, es mas universal que el acceso web y más ubicuo -no tienes que estar delante de un ordenador; desde tu movil, en un parada de autobús, puedes consultar las notas.

En este proyecto fin de carrera se modificará una aplicación CTI-IVR realizada por un antiguo alumno de la UPCT, incorporando un sintetizador de voz, que permita la consulta telefónica de notas de una forma más amigable.

1.2 ¿Qué es CTI?

CTI son las siglas de *“Computer Telephony Integration”*, su traducción directa nos da una idea bastante exacta de su significado, “integración de ordenador telefonía”.

Los sistemas CTI controlan interfaces de telefonía a través de aplicaciones permitiendo programar tareas y a través de ellos acceder a sistemas de otra tecnología, como puede ser el control de sistemas mecánicos desde un terminal telefónico. En sí, un sistema CTI nos permite interactuar, mediante llamadas telefónicas, con sistemas informáticos para realizar alguna tarea y viceversa. Los dispositivos telefónicos e informáticos más comunes sobre los que actúa un sistema CTI se detallan a continuación.

- Centralitas o PBX (*Private Branch eXchange*): La centralita suele ser el dispositivo principal del 90% de los sistemas CTI. Las PBX modernas

permiten todo tipo de conexiones al exterior: accesos básicos y primarios RDSI, líneas analógicas, líneas GSM e, incluso, conexiones de Internet (teniendo en la centralita o conectado a ella un módulo de voz sobre IP). La centralita hacia adentro ofrece líneas analógicas y digitales (que suelen tener un interfaz propietario del fabricante), también interfaces S0 (acceso básico RDSI) y terminales inalámbricos DECT (*Digital European Cordless Telephone*). Por último, lo que más nos interesa de una centralita son sus capacidades CTI. Es importante saber que una centralita sólo va a funcionar según los estándares (TAPI, etc.) si el fabricante ha incluido en ella el *"firmware"* adecuado. Muchas veces para la funcionalidad CTI es necesario adquirir un módulo adicional e instalarlo.

- Tarjetas Propietarias de Telefonía: son dispositivos pensados para CTI. Se conectan al ordenador por un puerto serie o directamente al bus ISA o PCI. También se conectan a una (o varias líneas telefónicas) para realizar alguna función específica. Por ejemplo: el dispositivo TAU-D del sistema MD110 de Ericsson **[1]** era una solución propietaria (ya obsoleta) que servía para crear un puesto de operadora basado en PC; las tarjetas de Dialogic **[2]** permiten conectar varias líneas telefónicas y realizar diversas funciones CTI avanzadas. Las tarjetas propietarias suelen ser caras y suponen atarse a la tecnología de un fabricante por lo que si se busca una solución abierta no son la mejor idea.

- Módems: es el dispositivo CTI más sencillo pero con una adecuada programación se puede lograr que haga funciones similares a algunas tarjetas propietarias de precio muy superior. Hoy día un buen módem debe tener capacidades de voz (estos módems llevan una tarjeta de sonido integrada, son capaces de emitir audio por la línea y de grabar lo que se recibe por ella, otras funciones como síntesis o reconocimiento habría que realizarlas por software dentro del PC), a veces, la falta de la capacidad de voz se puede suplir utilizando una tarjeta de sonido (conectándola al módem si tiene los conectores audio-in, audio-out). Otra capacidad interesante es que el módem sea capaz e reconocer los tonos DTMF (señales de numeración) que reciba por la línea. Hoy día cualquier módem es capaz de enviar y recibir faxes usando los estándares del grupo III y IV pero si queremos programar un envío de fax, será el software el que tenga que convertir el documento al formato

grupo III o IV (que son un estándares de compresión de imágenes blanco y negro binivel, esto es: sin niveles de gris intermedios). Por último, la capacidad más interesante (y más difícil de conseguir en un módem) es el reconocimiento del número llamante (para ello, por supuesto, es necesario que el operador incluya en la línea la señal correspondiente, muchas veces si el módem está conectado a una línea analógica de una centralita no es posible pasarle dicha señal).

- Teléfonos con Conexión a PC Incorporada: algunos fabricantes (como, por ejemplo, Selta **[3]**) proporcionan estos modelos que siempre hay que conectar a la línea telefónica y, opcionalmente, también al ordenador. Desde el ordenador se pueden hacer diferentes acciones como marcar, contestar (activando el manos libres), configurar el teléfono...

Sobre los dispositivos informáticos podemos hacer un par de comentarios acerca de las arquitecturas hardware/software utilizadas:

- Modelo *"Third Party CTI"*: a este modelo se le suele llamar "acceso en tercera persona", también podríamos llamarlo "modelo cliente-servidor". Debido a que es un modelo cliente-servidor hay un ordenador que realiza la tarea de servidor CTI y gestiona los recursos de telefonía, cuando cualquier otro desea realizar una tarea ha de conectarse a este a través de una red. Aunque los primeros sistemas de este tipo funcionaban sobre protocolo IPX (red Novell), actualmente se ha evolucionado hacia el protocolo IP. Concretamente son servicios de red que utilizan el transporte simplificado UDP (*User Datagram Protocol*) por lo que no son fiables fuera de las redes locales o, como mucho, metropolitanas. El ejemplo más importante de acceso *"Third Party"* es el estándar TSAPI (*Telephony Services Application Programs Interface*) que fue desarrollado conjuntamente por Novell y ATT.
- Modelo *"First Party CTI"*: a este modelo se le suele llamar "acceso en primera persona". Se trata de que cada ordenador que vaya a realizar funciones CTI esté conectado a uno o varios dispositivos telefónicos. En un modelo *"First Party"* puro, un ordenador sólo puede acceder a los dispositivos que están conectados a él (en *"Third Party"* cualquier ordenador puede acceder a cualquier dispositivo, sujeto a la política de permisos que se establezca en el servidor). Actualmente, existen modelos híbridos, esto es:

sistemas *"First Party"* que tienen una "extensión remota". La "extensión remota" es un módulo cliente-servidor que permite que un ordenador acceda a los dispositivos conectados a otro, como si fuera él el que los tiene conectados físicamente. Sin embargo, esta es una solución bastante pobre: da problemas a la hora de que dos ordenadores accedan a la vez a un mismo dispositivo y, desde un punto de vista de diseño, la "extensión remota" no es más que un remiendo sobre el sistema inicial. El ejemplo más importante de acceso *"First Party"* es el estándar TAPI (*"Telephony Application Programs Interface"*) que fue desarrollado por Microsoft, las versiones iniciales eran *"First Party"* puro, a partir de la versión 2.1 se incluyó el RTAPI (Remote TAPI).

También habría que hablar de los tipos de interconexión entre sistemas informáticos y telefónicos (enlaces CTI). Podríamos clasificarlos en dos tipos:

- Enlaces de datos: el enlace de datos más utilizado es un cable RS-232 (o enlace serie). Por ejemplo, las centralitas de gama media y alta de Selta tienen dos puertos serie. Uno de ellos debe usarse obligatoriamente para configurar la centralita desde ordenador en la primera instalación y en las acciones de mantenimiento (con una aplicación propietaria de Selta). El otro puede configurarse para alguna de estas funciones: Puerto de tarificación: el puerto de tarificación sirve para obtener información de todas las llamadas salientes de la central. Después de Finalizar cada llamada, se envía una trama con información que permite calcular el coste de la misma (número destino, fecha y hora de inicio, fecha y hora de fin, línea de salida, prefijo de acceso indirecto a operadores...). Puerto de Comunicaciones CTI: ese es el nombre que le da el fabricante. Evidentemente, el puerto de tarificación también es un puerto CTI (incluso el de configuración lo es). Este puerto sirve para conectar la centralita a un servidor de red. Estas centralitas utilizan el estándar TSAPI (acceso *"Third Party"*).Otros ejemplos de sistemas que usan enlaces de datos serían: las centralitas Kathrein se comunican con un ordenador usando un puerto serie pero el estándar utilizado es TAPI (sólo ese ordenador puede realizar acciones de control, a no ser que se tenga RTAPI), los módems (tanto si son externos como internos, el ordenador los ve como un dispositivo conectado por puerto serie), los teléfonos con conexión a PC (TAPI por puerto serie).

- Líneas Telefónicas: esta opción es menos habitual pero, a veces, los fabricantes obligan a establecer el enlace con una línea de la PBX. Por ejemplo las centrales de gama alta de Bosch admiten TSAPI pero la conexión con el servidor de telefonía es a través de un puerto S0 que hay que definir en la centralita. Para usar este esquema es necesario instalar en el servidor una tarjeta RDSI.

1.3 ¿Qué es IVR?

1.3.1 Introducción

IVR, Repsuesta Interactiva de Voz, es una tecnología madura que ayuda a miles de empresas en el mundo a atender llamadas telefónicas de manera automática y a consultar o manipular bases de datos y proporcionar información en forma de voz. El IVR permite que la información que se encuentra en sus servidores se encuentre disponible para el público que lo requiera.

1.3.1 Componentes de un sistema IVR

Podemos definir 3 componentes básicos en un sistema IVR, sin tener en cuenta las líneas telefónicas o la central PBX:

- CPU en el que corre la aplicación IVR.
- Interfaces de telefonía para conectar la CPU a la red de conmutación de circuitos (PSTN).
- Software que implementa el servicio del sistema.

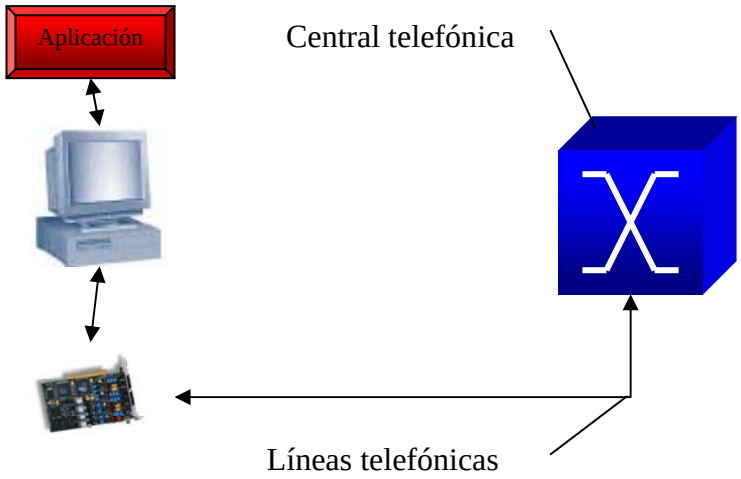

Figura 1.1: Esquema general de los componentes de un sistema IVR

La CPU puede ser un PC genérico o un computador de propósito especifico, esto nos puede limitar a la hora de compatibilizar la aplicación en distintas maquinas y del lenguaje y compiladores a utilizar.

1.3.2 Servicios IVR

Los servicios que se pueden ofrecer con los sistemas IVR quedan limitados por la información que se puede intercambiar a través de una llamada telefónica.

El tipo de usuario que interacciona con el sistema IVR puede ser una persona o una máquina.

En el caso de que el sistema opere con personas, estas pueden introducir los datos de dos maneras: mediante la pulsación de teclas o mediante reconocimiento de voz.

Si el método de introducir los datos es la pulsación de las teclas del teléfono, tenemos 12 dígitos para combinarlos y generar cadenas que tengan un significado, como puede ser un usuario a través del número de DNI, también podríamos utilizar sistemas de texto predictivo, igual que los SMS de los teléfonos, para acceder a mensajes con un vocabulario mayor y más sencillo para el usuario que una cadena de dígitos.

Si utilizamos sistemas de reconocimiento de voz podríamos ampliar el abanico de posibilidades utilizando diccionarios para la introducción de información y así proporcionar servicios más complejos, como el dictado de mensajes para después manipularlos, almacenarlos o transmitirlos.

En cualquiera de ambos métodos el sistema utiliza DSP’s para el análisis de la información introducida y para la reproducción de mensajes de voz grabados para

comunicarse con el usuario o las técnicas de text-to-speech ya existentes para leer mensajes de texto.

Si el sistema IVR se comunica con una maquina, por ejemplo una PBX, ambos utilizan protocolos de señales para realizar la comunicación entre ellos como los tonos de estado de las centrales telefónicas.

En base a estos métodos de comunicación los servicios que pueden ofrecer los sistemas IVR pueden ser muy diversos dependiendo de las funciones que la tarjeta sea capaz de realizar, como reconocimiento de dígitos DTMF, protocolos de FAX, protocolos de telefonía, grabación de mensajes, etc.

Algunos ejemplos de servicios son:

- Operadora automática.
- Buzón de mensajes.
- Reserva de billetes,
- Realización de encuestas.
- ACD "*Automated Call Distribution*".
- Información de servicios.
- Confirmación de servicios.
- Modificación de bases de datos.
- Gestión remota de sistemas.

La combinación de varios servicios en una aplicación da lugar a sistemas potentes que son capaces de realizar tareas de manera más eficiente que personas y con unos gastos muy inferiores.

1.4 Verbio TTS (Text to Speech- Texto a voz)

1.4.1 Introducción

La amplia penetración de la telefonía en la sociedad (impulsada aún con más fuerza desde la aparición de la telefonía móvil) obliga a la multitud de organizaciones a gestionar un gran volumen de llamadas (atención al cliente, campañas de marketing,

etc.). La posibilidad de automatizar la atención de gran parte de estas llamadas llevó a la aparición de hardware específico: las tarjetas de telefonía, que incorporan interfaces para poder interactuar con cualquier protocolo telefónico, proporcionando un gran abanico de funcionalidades destinadas a procesar cualquier tipo de llamada. Esta nueva tecnología que explota la interacción entre los mundos de la informática y de la telefonía se conoce, como ya hemos visto, con el nombre de Computer Telephony Integration (CTI).

Por otro lado, la constante evolución de los sistemas de reconocimiento del habla (ASR) y de síntesis del habla (TTS) ha permitido su incorporación en sistemas automáticos de recepción y emisión de llamadas telefónicas. Estos sistemas, que debido a su naturaleza eminentemente oral debieran haber sido el hábitat natural de las tecnologías del habla, recurrieron en sus inicios a estrategias tales como la reproducción de mensajes pregrabados y el reconocimiento por pulsación de tonos DTMF para interactuar con los usuarios. A pesar de que estas estrategias siguen siendo válidas e incluso aconsejables en algunos casos, actualmente las prestaciones obtenidas mediante la utilización conjunta o separada de sistemas ASR y/o TTS permiten dotar a los sistemas automatizados de una mayor naturalidad y fluidez en el proceso de intercambio de información con el usuario llamante.

La aparición en el mercado de una gran variedad de tarjetas telefónicas que se adaptan a cualquier necesidad y la evolución de los computadores actuales en cuanto a potencia de cálculo, han permitido que la tecnología CTI no requiera grandes inversiones y que, por lo tanto, también esté al alcance de las pequeñas y medianas empresas. Además, esta tecnología es fácilmente escalable, por lo que su implantación puede hacerse progresivamente en función de las necesidades de cada momento.

El desarrollo de Verbio **[4]** ha tenido en cuenta que la mayor parte de las aplicaciones actuales de los sistemas de reconocimiento y síntesis del habla están desarrolladas para funcionar en entornos telefónicos. Es por ello que el sistema de reconocimiento incorporado en Verbio ha sido entrenado con señales de audio procedentes de entornos telefónicos, tanto fijos como móviles, con el objetivo de obtener las mejores tasas de reconocimiento posibles en este tipo de entornos.

Verbio ha sido utilizado con éxito por integradores de todo el mundo trabajando sobre tarjetas de los siguientes fabricantes:

- Dialogic (Intel)
- Elcon
- NMS
- Avaya IR (Conversant)
- Altitude Software (Evox)
- Teima

Verbio permite ser utilizado conjuntamente con tarjetas CTI de otros fabricantes mediante los entornos de programación Library SDK **[5]** o Advanced SDK **[6]**. En ambos casos, el único requisito es que el desarrollador conozca suficientemente el API de programación de la tarjeta como para poder implementar toda la parte de adquisición y envío de muestra de/hacia la tarjeta CTI (intercambio de muestras con el sistema de reconocimiento de voz y síntesis de voz respectivamente).

1.4.2 Características principales

Verbio TTS es un sistema multilingüe de síntesis de voz que convierte, de forma automática, un texto escrito en una locución de voz natural, con la máxima inteligibilidad y entonación. Verbio TTS representa un importante salto cualitativo en la síntesis y le permitirá disponer de voces muy naturales en distintos idiomas.

Su objetivo es leer un texto con la máxima naturalidad y como la calidad se mantiene, y por lo tanto es controlada, es una herramienta imprescindible en aplicaciones cuyo texto, a priori, es desconocido o muy cambiante. Con la integración de Verbio TTS se evita la grabación constante de nuevos mensajes y se consigue un ahorro elevado en costes y tiempos, disponiendo de una flexibilidad total. Sus características principales son las siguientes:

- Voces de hombre y mujer de gran calidad, basadas en locutores profesionales.
- Entorno multilingüe con el que podrá sintetizar voces en español, catalán, euskera, gallego y otras lenguas francas internacionales como inglés, francés, portugués europeo y brasileño o mejicano.

- Entonación dinámica y adaptativa: selección de la curva de entonación original del locutor que mejor se adapta a cada contexto, superando la monotonía de los patrones estáticos.
- Algoritmo de ritmos y pausas del habla basado en la propia sintaxis y en métodos estadísticos.
- Verbio TTS tiene una amplia compatibilidad con distintos entornos de trabajo, que van desde los sectores de call centres, domótica, seguridad, portales de voz de servicios, aplicaciones de PC, aplicaciones industriales, móviles, PDA´s, etc.
- Locutor corporativo: podremos disponer de la voz corporativa que deseemos y que complemente los servicios de atención al cliente de forma natural. Además, conseguiremos una calidad de servicio superior ya que se desarrollará específicamente para las particularidades de nuestro sector en cuanto a entorno gramático, lingüístico y de vocabulario.
- Compatible con SAPI 4 y SAPI 5.

1.4.3 Aplicaciones

Verbio TTS está, especialmente, pensado para facilitar las comunicaciones personales, mensajes pregrabados, servicios de información, servicios transaccionales, otros servicios más específicos, etc.

1.5 Nuestra aplicación IVR

1.5.1 Escenario

Como comentamos, en un sistema CTI sus componentes básicos son:

- CPU en el que corre la aplicación IVR.
- Interfaces de telefonía para conectar la CPU a la red de conmutación de circuitos (PSTN).
- Software que implementa el servicio del sistema.

Además de estos componentes, aunque no son parte directa del sistema CTI-IVR, también están las líneas telefónicas y la central telefónica a la que están conectadas. Conocer que marca y modelo de central a la que el sistema esta conectado es muy importante, ya que el sistema debe interactuar con ella y para ello debe conocer los códigos de funciones, facilidades y las señales de señalización de los estados de la llamada, como por ejemplo el código de transferencia de llamadas utilizado en nuestra aplicación.

Los componentes del sistema que vamos a utilizar en nuestro escenario, estos son:

- La central PBX de la universidad es una central digital Alcatel que se encuentra conectada a la red pública mediante enlaces digitales primarios. Esta proporciona extensiones digitales propietarias y analógicas con rango de numeración público y marcación interna a 4 cifras. Ella será la encargada de proporcionarnos las extensiones que servirán como líneas para conectarlas a la tarjeta telefónica, y además nos realizara las funciones de encaminamiento de una llamada a una extensión libre de las cuatro.

- 4 líneas analógicas DTMF abonados de la PBX de la universidad, estas líneas cumplen las especificaciones del CCITT (UNE-CTR-21 *Reglamentación Técnica Común*) en cuanto a líneas analógicas y protocolos de señalización MF y DTMF.

- Un PC de propósito genérico, que cumple los requisitos hardware de la tarjeta de interfaces telefónicas que se va a conectar y que tiene instalado el software de gestión de la interfaz telefónica (API), herramientas de desarrollo y aplicaciones necesarias como MS Access, etc..

- Tarjeta de interfaces telefónicas, que ha sido proporcionada por el departamento TIC, del fabricante Dialogic modelo D4/PCI, la cual tiene un bus PCI para la conexión en el PC y 4 interfaces analógicas FXO *"Foreign Exchange Office"* para la conexión de líneas telefónicas analógicas de una central.

- Aplicación CTI-IVR que fué desarrollada en lenguaje C utilizando el API ODBC de Microsoft y el API de Dialogic, en el entorno de programaciónDev-C++.

- Soporte de licencias para la conversión texo a voz, que ha sido proporcionada por el departamente TIC, del fabricante Verbio, el cual tiene un puerto USB para su conexión con el PC.

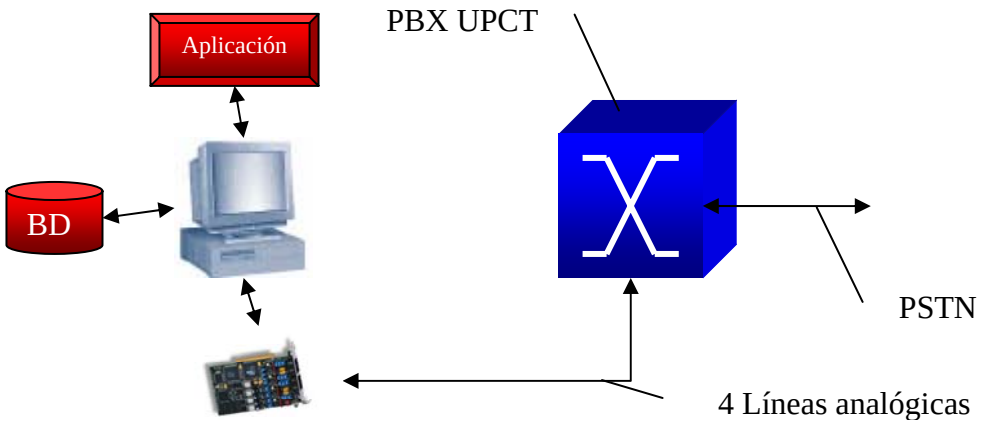

Figura 1.2: esquema del sistema diseñado

1.5.2 Descripción de Componentes

Ahora profundizaremos más acerca de cada componente, dejando la parte de instalación y configuración para el capitulo 3 en el que se detallará la puesta en funcionamiento y el mantenimiento del sistema para el administrador.

Tarjeta de interfaces telefónicas

Para poder plantear una aplicación IVR, como hemos dicho antes, debemos de conocer las capacidades del hardware que disponemos. En nuestro caso disponemos de una tarjeta de voz del fabricante Dialogic modelo D4/PCI**,** esta tarjeta Dialogic la cataloga dentro de su familia *Voice Board*, que son tarjetas de voz capaces de analizar las señales del canal de telefonía, pero no disponen del hardware para el manejo de señales de fax ni el procesado de voz en tiempo real, no disponen de CSP *Continuous Speech Procesor* que permite realizar llamadas de voz full dúplex.

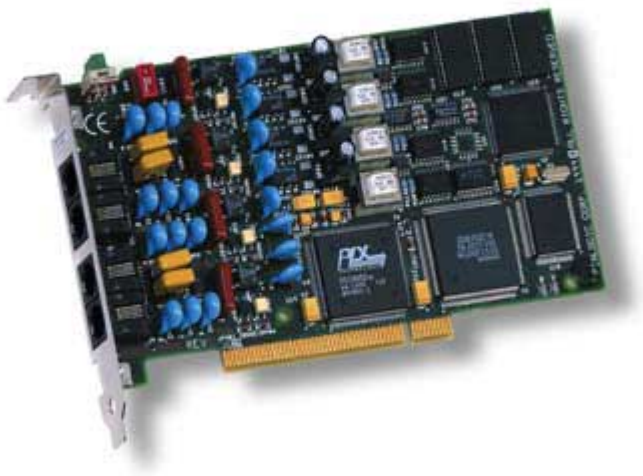

Figura 1.3: Fotografía de la tarjeta Dialogic D4/PCI.

Para consultar más información acerca de sus características técnicas podemos consultar la hoja de características de la tarjeta **[7]**.

La tarjeta que disponemos es sin duda una de las tarjetas ideales para el desarrollo de una aplicación IVR avanzada dado a sus capacidades de manejo de las líneas analógicas y su DSP que nos permite el análisis complejo de las señales y protocolos necesarios para utilizar y manejar tanto líneas analógicas como centrales PBX. Dialogic nos dice que esta tarjeta es la ideal para sistemas CTI-IVR de tamaño medio, ya que en un mismo sistema físico se pueden conectar hasta 64 puertos añadiendo tarjetas. Además todos sus parámetros son configurables lo que permite añadir señales de supervisión que no hayan sido implementadas en sus firmware y configurar esta para los distintos protocolos de telefonía de otros países que tengan variaciones en cuanto a frecuencias de señalización o a tiempos de señales.

El software que proporciona Dialogic tiene varias herramientas y utilidades, además del API para la programación de aplicaciones con la tarjeta y una documentación bastante completa de programación y desarrollo de aplicaciones incluyendo ejemplos de aplicaciones básicas. Esta documentación será detallada en el

anexo de Dialogic. También hay que comentar que el *System Release* de Dialogic proporciona 3 aplicaciones muy útiles:

- *PBXpert*, para la configuración de los tonos de señalización de la PBX a la que se conecta la tarjeta y funciones TAPI.
- *Configuration Manager* (DCM), para la inicialización y configuración de las tarjetas que haya en el sistema.
- *Universal Dialogic Diagnost Utility*, para el diagnostico de la tarjeta, comprueba todo el hardware y realiza test de estado presentando un informe.

Esta tarjeta nos permite realizar cualquier aplicación IVR e incluso la ampliación del sistema, ya que permite hasta 64 puertos dentro de un mismo PC. Los requisitos de la tarjeta son:

Hardware: 80486, Pentium o compatible con Intel.

PC del sistema

El PC utilizado es un Pentium IV a 2.4 GHz con 256 MB de memoria RAM y un disco de 20GB. El sistema operativo elegido es Windows XP, ya que uno de los requerimientos del *System Release* de Dialogic **[8]** de la tarjeta es Windows NT, 2000 o XP.

PBX

El sistema de telefonía de la Universidad Politécnica de Cartagena se compone por dos PBX marca Alcatel modelo 4400 V3.2 conectadas entre si por dos primarios propietarios de Alcatel (primarios abc) permitiendo entre ellas 60 comunicaciones simultaneas, una de estas PBX se encuentra físicamente en el edificio de la Avda. Alfonso XIII la cual esta conectada a la red publica por un primario RDSI sobre fibra óptica (30 canales de voz) con rango de numeración público, la otra PBX se encuentra físicamente en el edificio del hospital de marina y esta conectada a la red pública mediante un primario RDSI sobre modems HDSL a movistar, están configuradas para que cualquier extensión de cualquiera de ellas encamine la llamada por el primario que más interese (las llamadas a la red fija por el primario de telefónica España, las llamadas a la red móvil por el primario de telefónica movistar), además existe una red corporativa que reduce el costo de las llamadas entre los abonados de ella permitiendo además una marcación dentro de la red corporativa a 4 cifras tanto para telefonía fija como móvil.

Este sistema telefónico nos proporcionará las cuatro extensiones analógicas para conectarlas a la tarjeta Dialogic.

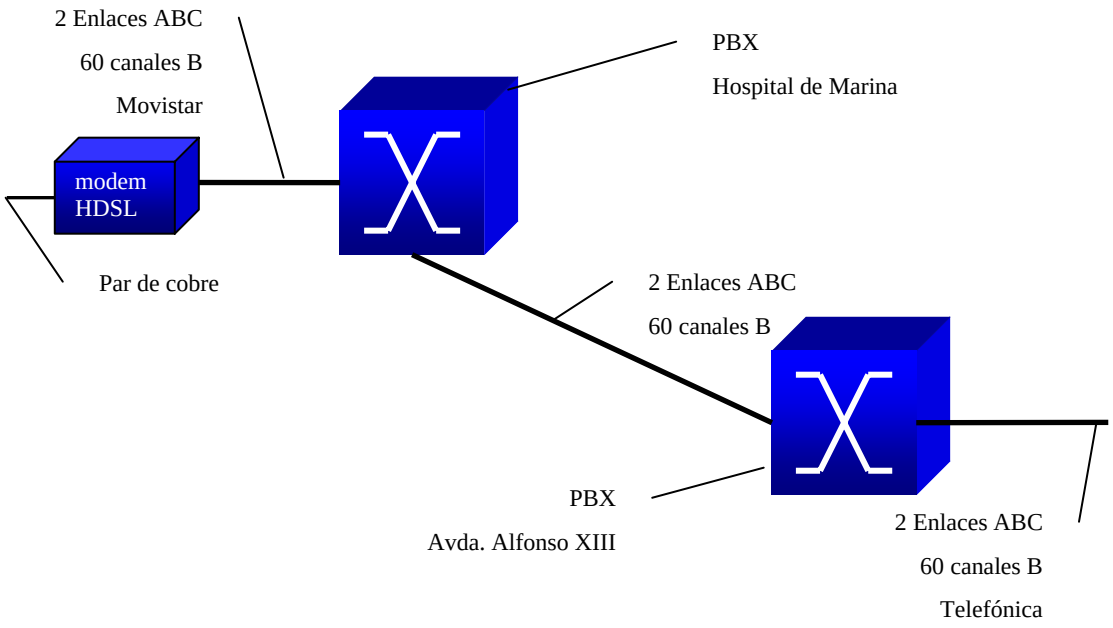

Figura 1.4: Arquitectura del sistema de telefonía de la UPCT

Líneas telefónicas

Este sistema telefónico nos proporcionará las cuatro extensiones analógicas para conectarlas a la tarjeta Dialogic y al menos un número de teléfono de la red publica el que será el número de cabecera de un grupo de extensiones para la distribución de las llamadas entrantes. La PBX encamina una llamada entrante al número de cabecera, hacia una de las cuatro extensiones que este libre de acuerdo a un algoritmo programado en la central (lineal, cíclico o circular, esta distribución no es importante se puede optar por cualquiera de ellas). Las extensiones deben de permitir la marcación multifrecuencia y tener el servicio de transferencia de llamadas habilitado. No es necesario ni restrictivo que tenga algún otro servicio habilitado.

Aplicación

La aplicación fué desarrollada en C, y seguirá en C ya que el API de Dialogic esta implementado en este lenguaje y el API de Verbio incorporado también. Dialogic nos

proporciona los archivos de cabecera y las librerías estáticas y dinámicas. Dialogic no proporciona bastante información acerca de la tarjeta y ayuda para desarrollar aplicaciones con ella, esta información esta en varios documentos que serán comentado en mayor profundidad en el apéndice del API de Dialogic, estos documentos son:

- ***1828-02.pdf***. Describe el contenido del Service Pack 5.1.1 para Windows.
- ***dnacfgp.pdf***. Guía de configuración e instalación.
- ***featgd3.pdf***. Guía de características.
- ***icdcust.pdf***. Personalización de las herramientas de configuración
- ***Installation Guide.pdf***. Guía de instalación.
- ***pgmgd3.pdf***. Guía del programador de Windows.
- ***pt_history.pdf***. Lista histórica de puntos y problemas resueltos.
- ***release_guide.pdf***. Guía del System Release 5.1.1 de Windows.
- ***release_update.pdf***. Actualización del System Release 5.1.1 de Windows.
- ***srlgd3.pdf***. Guía de la librería estándar de runtime de Windows.
- ***userguide.pdf***. Guía del usuario de Windows.

A su vez, Verbio también proporciona información de cómo desarrollar una aplicación con VerbioTTS, esta información está contenida en varios documentos que comentarán más profundamente en el apéndice de Verbio:

- ***guide_es.pdf.*** Documentación necesaria para la instalación, configuración, etc.
- ***Dialogic_sdk_es.pdf.*** Descripción de todo el API de Dialogic con algunos ejemplos.

El entorno de programación a utilizar es un entorno libre con licencia GNU llamado Bloodshed Dev-C++ **[9]** v4.9.8.0, este nos proporciona un entorno de programación con procesador de texto para lenguaje C y C++, compilador y herramientas de *debug*.

1.5.3 Capítulos de la Memoria

La memoria del proyecto se dividirá en cinco capítulos en los que se dará una idea de los sistemas CTI y se detallara la aplicación CTI-IVR desarrollada con unos manuales para usuarios y para el administrador del sistema, estos capítulos son:

- Capítulo primero: en el presente capitulo se han introducido los sistemas CTI, IVR y VerbioTTS explicando sus componentes, escenarios y servicios que proporcionan. También se ha introducido la aplicación desarrollada con sus componentes y el escenario particular
- Capítulo segundo: en este capitulo contiene un manual para el usuario, desde el punto de vista del alumno, explicando los requisitos para acceder al sistema y como el alumno tiene que utilizarlo, con una explicación de los estados complementada con un esquema funcional de la aplicación, y desde el punto de vista del profesor, cuál es su papel en este sistema.
- Capítulo tercero: en este capitulo se profundiza en el sistema para que el administrador del sistema sea capaz de instalar, configurar y mantener el sistema y como podrá controlar la aplicación, es decir, hacer un seguimiento de las consultas realizadas, cambiar su clave, inicializar la tabla de consultas, restablecer claves de alumnos, etc. también se detalla el diseño de la aplicación para introducir los conceptos necesarios para una modificación de la aplicación con poco esfuerzo.
- Capítulo cuarto: en este capítulo se explica la arquitectura del sistema, sus estados principales, su flujo de ejecución etc.
- Capítulo quinto: en este capitulo se detallan las conclusiones y se proponen líneas futuras de desarrollo.

1.5.4 Apéndices

Existen tres apéndices en los que se detallan las API’s que se han utilizado y se explica la estructura de ficheros de la aplicación, estos apéndices son:

- Apéndice A: Descripción del funcionamiento de un sintetizador de voz general.

- Apéndice B: Descripción del API de Dialogic detallando el contenido de los documentos que proporciona Dialogic.
- Apéndice C: Descripción del API de Verbio detallando el contenido de los documentos que proporciona Dialogic.

Capítulo 2: Manual de Usuario

2.1 Introducción

En este manual se proporciona una descripción del sistema CTI-IVR y del sistema VerbioTTS diseñado y utilizado en este Proyecto Fin de Carrera. El enfoque será el de generar un manual de consulta para el usuario, que describa las funcionalidades del sistema, sin profundizar en aspectos de implementación interna. Estos aspectos serán desarrollados en el capítulo 3.

El sistema CTI-IVR desarrollado es una aplicación la cual nos permite la consulta interactiva de las notas de las asignaturas cursadas en la convocatoria vigente, que se encuentran en una base de datos, mediante una llamada telefónica y la interacción con la aplicación desde el teléfono pulsando las teclas para así realizar la consulta de notas.

El software de Verbio aportará la propiedad de síntesis del habla, es decir una conversión de texto a voz. De forma que los mensajes de voz dados al usuario resulte más o menos y reales que si fueran mensajes previamente grabados.

La aplicación corre en un PC que tiene una tarjeta conectada a líneas telefónicas. Ésta realiza consultas a la base de datos donde se guardan las notas. Una vez consultadas las notas, el PC reproduce las notas mediante un sistema de conversión de texto en habla, de la casa Verbio, al usuario que la escuchará por el auricular de su teléfono.

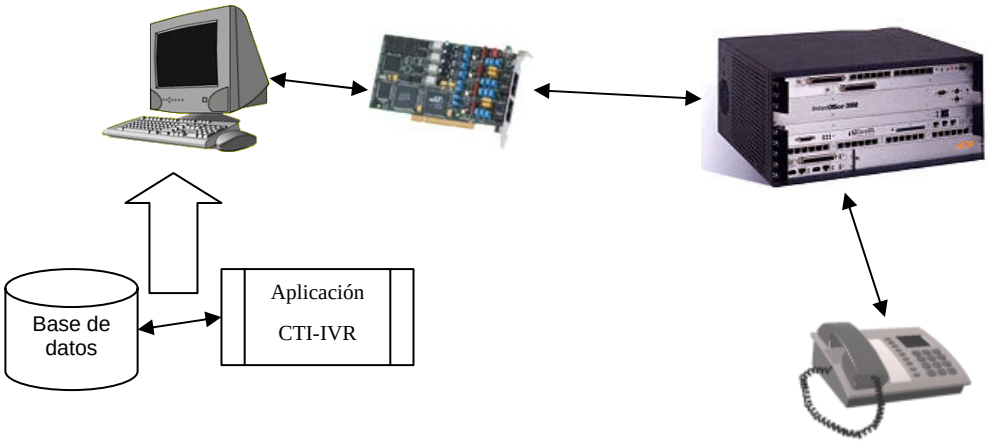

Figura 2.1: Esquema grafico del sistema

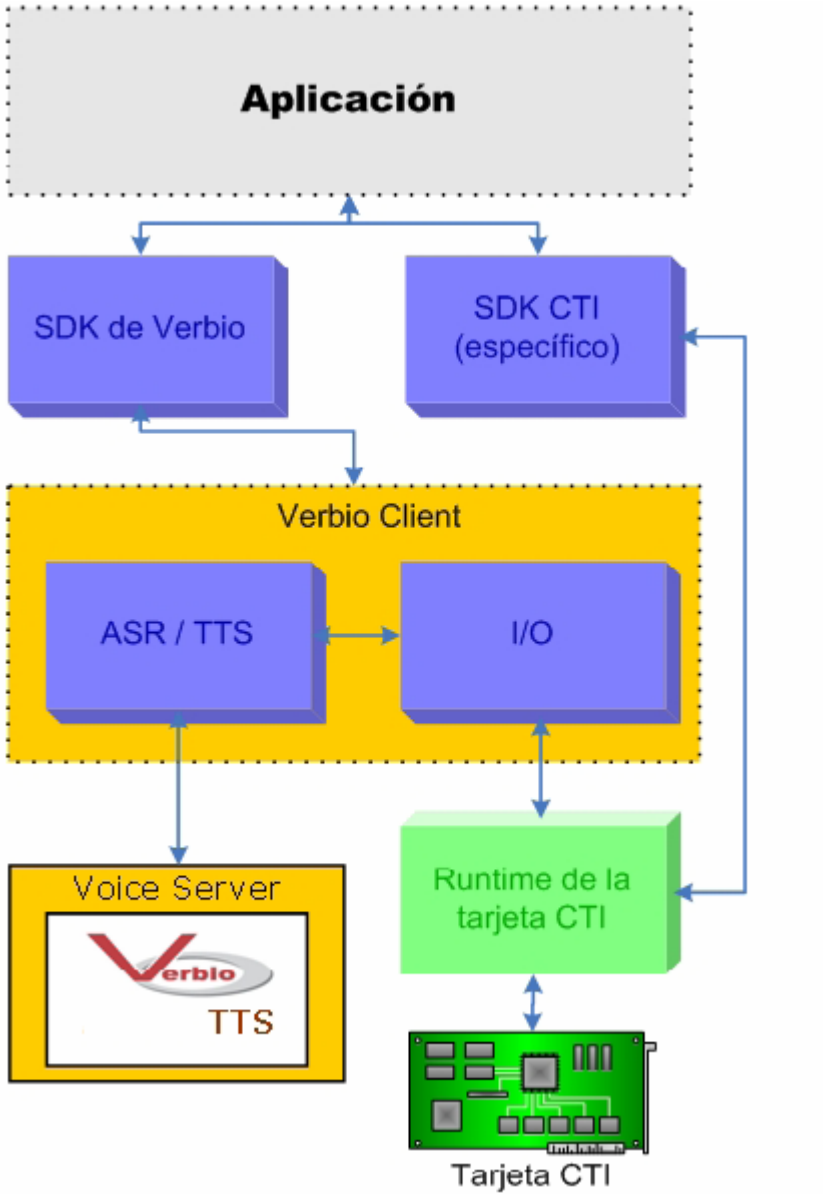

Figura 2.2: Arquitectura cliente-servidor del sistema.

El sistema informa y ayuda al usuario de los datos que va necesitando o proporciona la información solicitada en la consulta. El usuario accederá al sistema realizando una llamada al número de teléfono donde esta conectado el sistema y navegará por la aplicación mediante la selección de opciones o introducción de datos con las pulsaciones de las teclas de su teléfono, para ello deberá conocer algunos datos referentes a la consulta que desea realizar.

El usuario puede realizar las acciones de: (1) consulta de nota de asignatura introduciendo código de la asignatura, (2) consulta de notas de todas las asignaturas sin necesidad de introducir el código de las mismas, (3) cambiar su clave de acceso y (4) salir, dar por finalizada la consulta.

A continuación se especifican los requisitos que el usuario debe cumplir o conocer para usar el sistema, el esquema y las posibles acciones que puede realizar en la aplicación.

2.2 Requisitos de acceso al sistema

Para que el usuario pueda acceder al sistema debe utilizar un terminal telefónico capaz de producir tonos multifrecuencia (DTMF) con las pulsaciones de las teclas, esto es un teléfono que produce un tono distinto e identificativo para cada tecla.

A continuación debe de conocer el número de teléfono donde el sistema esta conectado, para realizar una llamada y que la aplicación atienda la petición de servicio.

Como las consultas son personales debe de conocer el código de usuario (DNI) y su clave de acceso.

Además, si quiere saber la nota de una asignatura específica y no la de todas las asignaturas que cursa y que ya estén disponibles para consultar, necesita el código de la asignatura que desea consultar, estos se pueden encontrar en la guía de los planes de estudios o bien, preguntando al profesor de la asignatura. También se debe señalar que las notas disponibles para consultar serán aquellas correspondientes a las asignaturas de la convocatoria vigente. El sistema informará al alumno de la convocatoria actual en el momento de su llamada, ésta podrá ser Febrero, Junio, Septiembre o Diciembre.

La aplicación permite realizar las siguientes acciones dependiendo de la opción que seleccione en los menús:

- Consulta de nota de una asignatura específica.
- Consulta de notas de asignaturas.
- Cambio de clave de acceso.
- Salir del sistema.

2.3 Esquema de la aplicación y detalle de las acciones implementadas

En esta sección describiremos el diagrama de flujo y la explicación de las acciones que el usuario debe realizar en cada estado por el que puede pasar.

Para realizar una consulta llamamos al número del sistema y la aplicación nos responderá con un mensaje de bienvenida, a continuación nos pedirá el DNI que introduciremos marcando desde el teléfono los 8 números del DNI, en caso de introducir un DNI no valido nos permitirá introducirlo de nuevo, si nos equivocamos tres veces la llamada finalizará. Seguidamente nos pedirá una clave de acceso. En un principio la clave de acceso de cada alumno se corresponderá con los ocho dígitos numéricos de su fecha de nacimiento de la forma que si nació el 21 de Julio de 1983, su clave será 21071983. Esta clave se recomienda que se cambie por otra personal y exclusiva para cada alumno, de manera que, si el alumno que usa la aplicación todavía no ha modificado la clave por defecto o es la primera vez que usa el servicio, escuchará un mensaje que le advertirá que, por motivos de seguridad, sería recomendable que cambiase su clave de acceso. Una vez autentificados, el sistema nos dará a elegir entre varias opciones, pulsando un número para seleccionar cada una de ellas. Las opciones son: consulta de nota de una asignatura específica, consulta de notas de todas las asignaturas de la convocatoria actual, cambio de clave de acceso o salir del sistema pulsando 1, 2,3 o 4 respectivamente. Estas opciones son detalladas a continuación:

- Si marcamos 1 (consulta de nota de una asignatura específica) el sistema nos indica que introduzcamos el código de la asignatura mediante el teclado del teléfono (este código lo consultamos en la guía docente) si el código es erróneo el sistema nos invita a introducirlo de nuevo (tres códigos erróneos provocan la desconexión de la llamada), si aún no se han introducido notas o

la asignatura no existe el sistema nos lo indicará con el correspondiente mensaje. Después de introducir el código de la asignatura correctamente y que las notas ya estén introducidas en la base de datos escucharemos un mensaje compuesto por el nombre de la asignatura y la nota alfanumérica que hemos obtenido en el examen de la asignatura correspondiente a ese código, seguidamente si también está introducida la nota numérica nos la dirá y si no, nos avisará de que no hay nota numérica. La nota alfanumérica podrá ser: Matricula de Honor, Sobresaliente, Notable, Aprobado, Suspenso, también se distinguirá entre si no hay notas de la asignatura o si el alumno no se ha presentado a la misma. A continuación nos dará a elegir entre dos opciones de otro menú: realizar otra consulta o salir del sistema pulsando 1, o 2 respectivamente.

  - Si marcamos 1 iremos al menú principal para volver a elegir opción.
  - Si marcamos 2 saldremos del sistema.

- Si marcamos 2 (consulta de notas) el sistema no nos pide nada más y pasará directamente a consultar todas las notas de las asignaturas de las que estemos matriculados en la convocatoria vigente, identificando las que tienen nota, las que no la tienen todavía y las que tienen nota pero no nos hemos presentado al examen, de la misma forma que en el caso anterior y con el mismo formato de respuesta. A continuación nos dará a elegir entre dos opciones de otro menú: realizar otra consulta o salir del sistema pulsando 1, o 2 respectivamente.
  - Si marcamos 1 iremos al menú principal para volver a elegir opción.
  - Si marcamos 2 saldremos del sistema.
- Si marcamos 3 (cambio de clave de acceso) el sistema nos indicará que introduzcamos la nueva clave dos veces, ambas deben de ser iguales. La nueva clave se modificará en la base de datos indicándonos con un mensaje si ha sido cambiada con éxito. A continuación nos dará a elegir entre dos opciones de otro menú: realizar otra consulta o salir del sistema pulsando 1, o 2 respectivamente.
  - Si marcamos 1 iremos al menú principal para volver a elegir opción.

    - Si marcamos 2 saldremos del sistema.

- Si marcamos 4 (salir del sistema) la aplicación nos despedirá con un mensaje de despedida y terminará con la llamada.

El esquema de la aplicación es el siguiente:

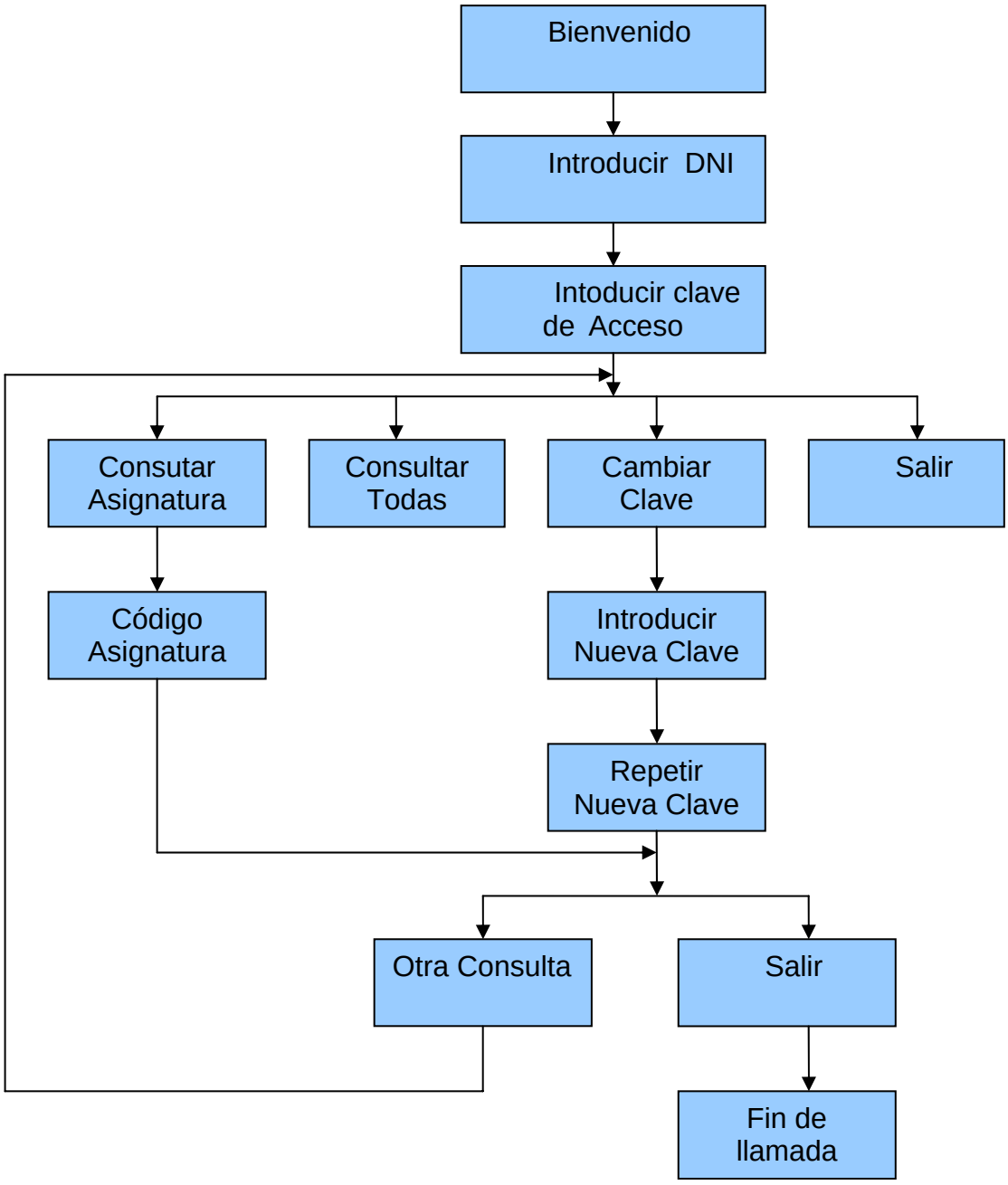

Figura 2.3: Esquema de la aplicación.

2.4 Usuario Profesor

A parte del alumno, también existe la figura del usuario profesor que aunque sólo disponga de un papel en esta aplicación es el más importante.

A día de hoy, el sistema de notas que usan los profesores es el siguiente:

1. Cuando han corregido los exámenes, sacan las notas provisionales en el tablón de anuncios y en muchos casos, también en la página web de la asignatura.
2. Proponen una fecha y hora para la revisión, normalmente suele ser una por la mañana y otra por la tarde, para que los alumnos que lo deseen puedan acudir a ver sus exámenes o reclamar.
3. Después de esta/s revisión/es los profesores sacan las notas definitivas, y es entonces cuando introducen las mismas en AGORA.

En Agora, se encuentran las bases de datos de Gestión Académica, por este motivo, para que este proyecto se pueda llevar a la práctica, es muy importante el compromiso del profesor a, no sólo introducir las notas definitivas si no que a hacerlo antes, con las notas provisionales y posteriormente modificarlo con las definitivas. Esto es, porque, como ya se comentó al principio esta aplicación está orientada a aquellos alumnos que no puedan o quieran desplazarse exclusivamente a la universidad, o que no dispongan de Internet, para consultar sus notas, y cuando el alumno consulta una nota le interesa saber que tiene una/s ocasión/es para ir a ver su examen y reclamar su nota antes de que ésta pase a formar parte de la Acta y ya no haya marcha atrás.

Es por este motivo por el que la figura del profesor es muy importante. Será necesario concienciar al profesorado y a pedirle su absoluta colaboración.

Capítulo 3: Manual de Administrador

3.1 Introducción

En este capítulo proporcionaremos una descripción en profundidad del sistema, destinada al usuario administrador. Describiremos de forma detallada el proceso de instalación, esquema de la aplicación, tarjeta IVR y su software y el software de VerbioTTS. Dividiremos este capítulo en cinco apartados en los que se detallarán cada una de las partes del sistema y sus características, estas partes son:

- Central telefónica a la que está conectada la tarjeta de telefonía. En este apartado se detallarán que requisitos debe de cumplir la central, que tipo de líneas se conectarán, como deben de configurarse las líneas y que servicios deben estar disponibles y cuales no.
- Tarjeta de telefonía. En este apartado se describirá como se instala la tarjeta y su software y las configuraciones de la tarjeta y del software con respecto a la central a la que está conectada.
- Sintetizador de voz VerbioTTS. En este apartado se describirá la inicialización, configuración y licencias del software.
- Bases de datos. En este apartado se describirá el tipo de base de datos utilizada, cuales son sus tablas y campos, como se instala las bases de datos en el sistema y como se mantienen.
- Aplicación desarrollada. En este apartado se describirá la aplicación que es el principal elemento de este proyecto, como se instala y configura, cual es su estructura, como se pueden añadir nuevas funcionalidades y estructura de ficheros.

3.2 Central Telefónica

En este apartado se describe para la central telefónica los requisitos que deben de cumplir las líneas telefónicas, los servicios que debe de proporcionar y los conceptos básicos para que el administrador del sistema pueda indicar al responsable del

mantenimiento de la central cuales son las necesidades del sistema para ponerlo en funcionamiento.

La central telefónica proporciona las líneas que se conectan al equipo para permitir las llamadas y así poder ofrecer un servicio IVR. Estas líneas pueden ser abonado de una central privada PBX o de una central publica PSTN (*Public Switched Telephone Network*), en ambos casos deben de ser líneas analógicas que cumplen la norma CTR-21 **[10].**

Líneas telefónicas

Las líneas telefónicas, independientemente de que pertenezcan a una central publica o privada, deben de ser extensiones analógicas que permitan la marcación multifrecuencia DTMF, deben de cumplir la norma CTR-21 y estar agrupadas en un grupo de llamada para que la central a la que pertenece distribuya las llamadas de forma eficaz y eficiente. Estas líneas serán conectadas a la tarjeta Dialogic a sus puertos correspondientes mediante un cable telefónico con un conector RJ-11 de un par trenzado de cobre a los pines centrales del conector, pines 3 y 4, para cada línea.

Figura 3.1: Vista de los conectores RJ-11 en la tarjeta Dialogic

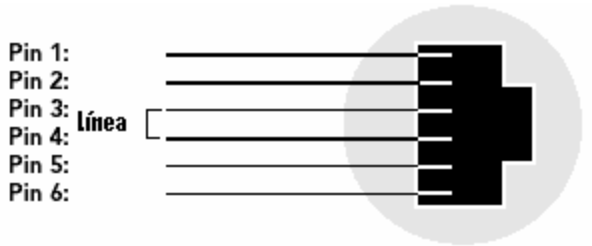

Figura 3.2: Conector RJ-11

3.2.1 Grupos de llamada

Un grupo de llamada es un servicio que las centrales digitales son capaces de ofrecer para distribuir llamadas a un solo número de teléfono entre varias extensiones con un algoritmo de distribución de llamadas programado. Al número de teléfono al que se debe de llamar para acceder al grupo de llamada se le llama número de cabecera. La distribución de las llamadas que las centrales pueden presentar como opción a un grupo de llamada son:

- General, la central señalizara la llamada en todas las extensiones agrupadas y la primera que atienda la llamada será conectada con el llamante.
- Lineal, la central señalizará una llamada entrante a la extensión que primero se ha añadido al grupo, si la extensión está ocupada o no disponible la llamada será señalizada a la segunda extensión del grupo y así sucesivamente.
- Cíclica, la central señalizara la primera llamada por la extensión que primero se añadió al grupo, la siguiente llamada que entre a la segunda extensión añadida, y así sucesivamente. Si se llega a la última extensión añadida se repetirá el ciclo. Si la extensión a la que le corresponda la llamada esta ocupada o no disponible la llamada será señalizada a la siguiente.

Para nuestra aplicación la distribución de llamadas entre el grupo de cuatro extensiones ha de ser preferentemente cíclica, siendo valida también la distribución lineal. La distribución general no es válida y debe de ser especificado al administrador de la central a la que conectemos el sistema que esta distribución no debe de configurarse en nuestro grupo de llamadas.

3.3 Tarjeta Dialogic D4/PCI

En este apartado se describe como instalar la tarjeta y su software y como se configura para que el administrador del sistema pueda realizar las tareas previas a la instalación y puesta en funcionamiento del sistema.

La tarjeta de telefonía es una tarjeta Dialogic modelo D4/PCI, esta es una tarjeta con un bus PCI para su conexión en la placa madre de un PC, dispone de 4 interfaces analógicas de telefónica con conectores RJ-11.

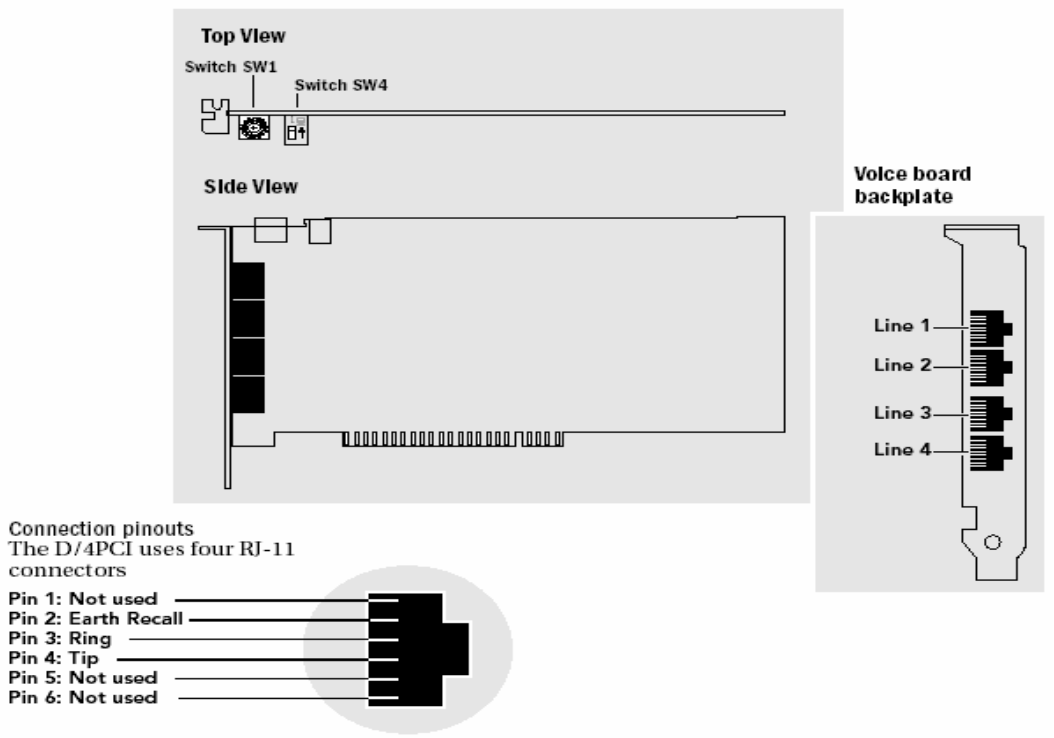

Figura 3.3: Dibujo de la tarjeta y detalle de sus conectores y switches

A continuación describiremos los procedimientos de instalación y configuración de la tarjeta y su software.

3.3.1 Instalación y configuración del hardware

Quitamos las tapas del PC para tener acceso al bus PCI, elegimos una ranura libre, si tenemos varias intentaremos dejar alguna libre entre nuestra tarjeta y otras para mejorar la ventilación, e insertamos la tarjeta Dialogic en la ranura. Colocamos el tornillo o pieza de sujeción de las tarjetas PCI y ya estamos listos para la configuración del hardware.

Figura 3.4: Fotografía de la tarjeta instalada en el PC (interior)

La tarjeta dispone de dos switches en la placa para la configuración del número de tarjeta y del estado de las líneas cuando la tarjeta no esta iniciada. Estos son el SW1 y el SW4 que se aprecian el la figura 3., el SW1 tiene 17 posiciones, la posición 0 indica que el número de tarjeta lo asigna el SO en función del slot PCI (geográficamente) y el número de tarjetas en el sistema y de la posición 1 a la F, 16 posiciones en hexadecimal, para seleccionar el número de tarjeta del sistema y asignar la numeración a los canales, el SW4 es un interruptor de dos posiciones ON y OFF que indica el estado OFFHOOK u ONHOOK, respectivamente, en el que las líneas estarán cuando la tarjeta no este iniciada y reciba una llamada por uno de sus canales. Después conectamos los latiguillos de las líneas telefónicas en los conectores RJ-11. Una vez que la tarjeta ha sido instalada en el PC lo iniciamos para que el SO reconozca la nueva tarjeta. Al iniciarse por primera vez el PC con esta tarjeta el asistente de nuevo hardware encontrado nos indica en una ventana que la tarjeta recién instalada ha sido encontrada.

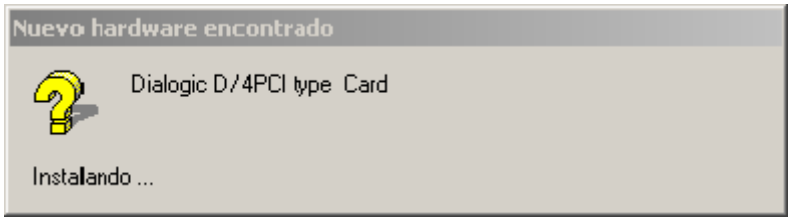

Figura 3.5: SO encuentra la tarjeta Dialogic instalada

Si la versión de SO es moderna instalara los ficheros .inf que dan información acerca de la tarjeta, pero aún así no están instalados aun los driver ni el software de ella.

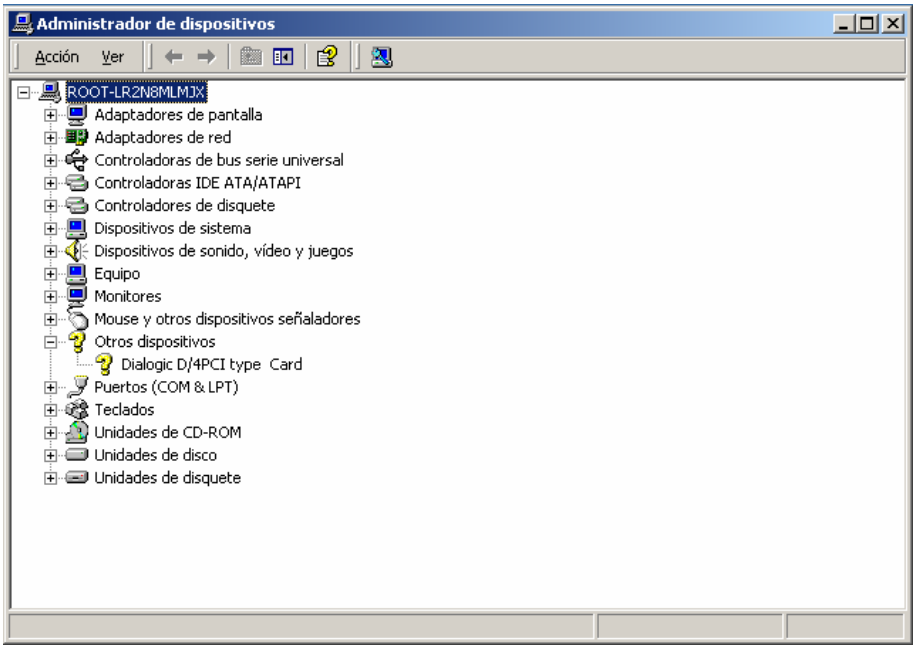

Figura 3.6: Administrador de dispositivos con la tarjeta recién reconocida

En este punto debemos de instalar el *System Release* que contiene los *drivers* y el software de la tarjeta para su configuración.

3.3.2 Instalación y configuración del software

Una vez concluida la instalación física de la tarjeta y reconocida por el SO, tenemos que instalar el software para poder utilizarla. El software de la tarjeta esta en un pack llamado *System Release*, actualmente Dialogic tiene la versión 6.0 para Windows y linux (Red Hat 7.3), este software no viene con la tarjeta y hay que descargarlo de la web de Dialogic **[12],** la aplicación se ha desarrollado con el SR 5.1.1 con SP1 ya que este se puede descargar gratuitamente y el SR 6.0 no. Una vez que disponemos de los dos ficheros para instalar el SR, SR511All.zip y SR511SP1.zip, los descomprimimos en el directorio raíz del disco duro, en otro directorio con una ruta más compleja puede dar problemas de instalación por la ruta, e instalamos primero el SR511All.zip ejecutando setup.exe.

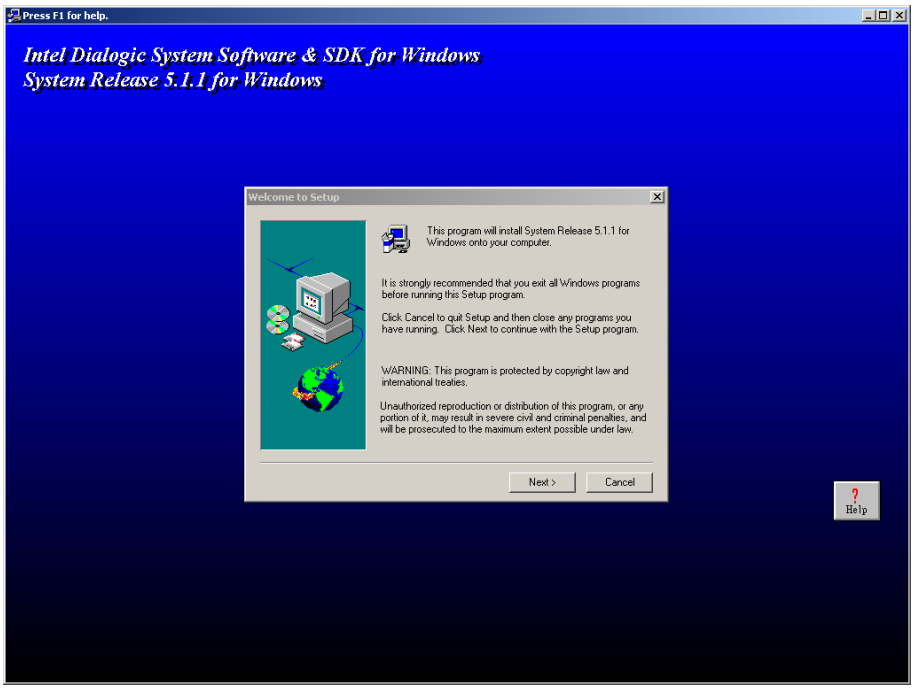

Figura 3.7: Instalación del SR 5.1.1 (I)

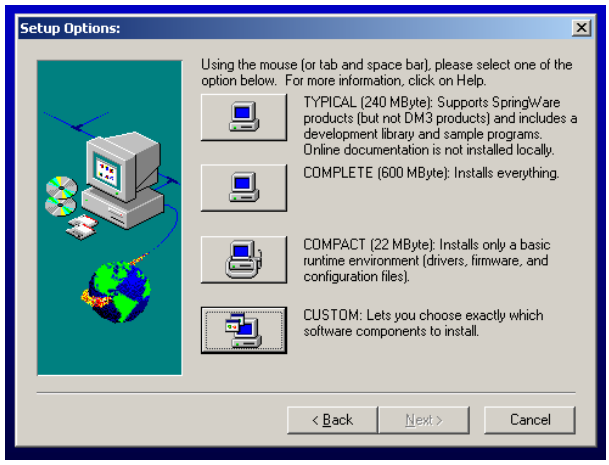

Figura 3.8: instalación del SR 5.1.1 (II)

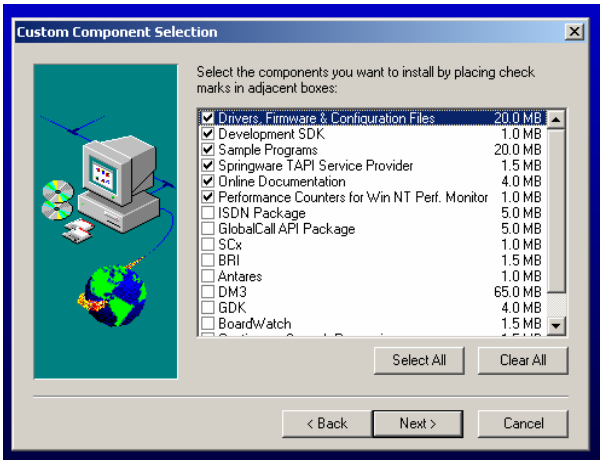

Figura 3.9: instalación del SR 5.1.1 (III)

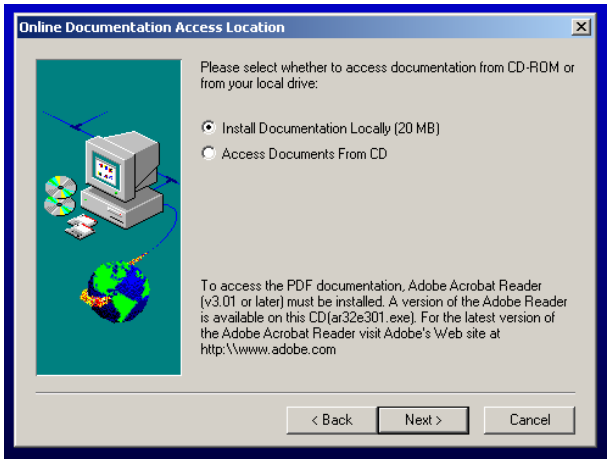

Figura 3.10: instalación del SR 5.1.1 (IV)

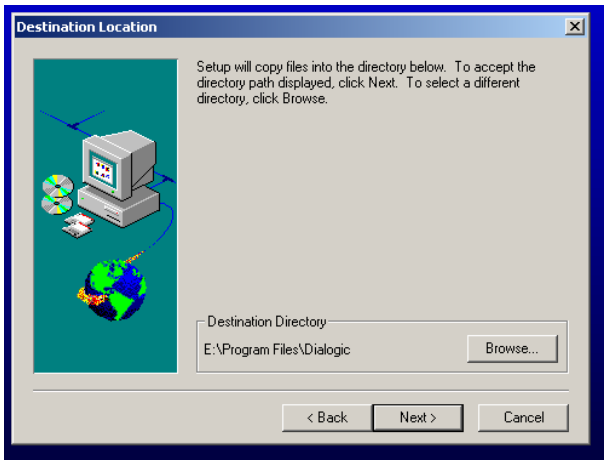

Figura 3.11: instalación del SR 5.1.1 (VI)

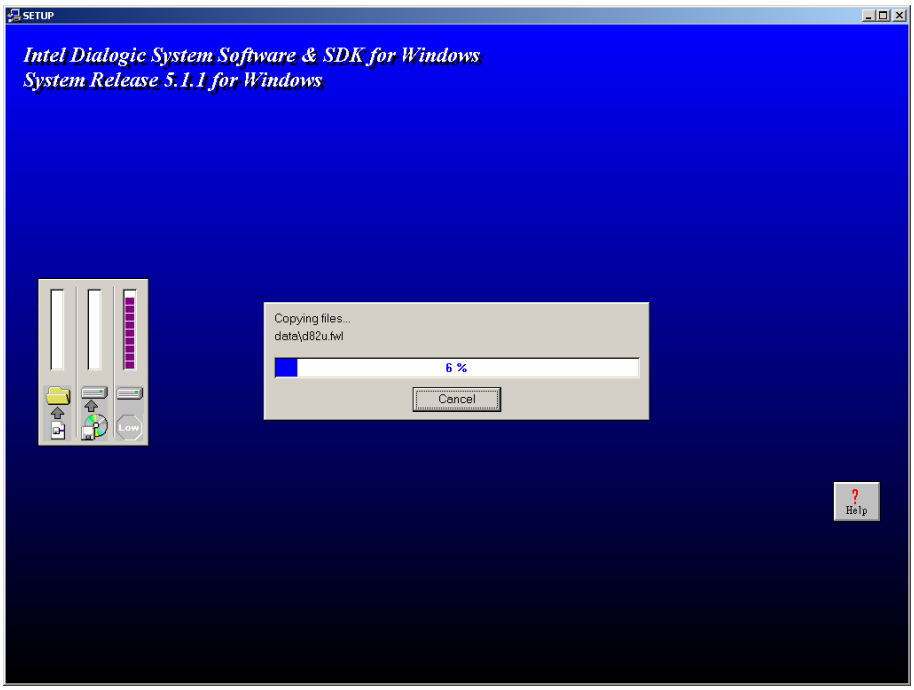

Figura 3.12: instalación del SR 5.1.1 (VI)

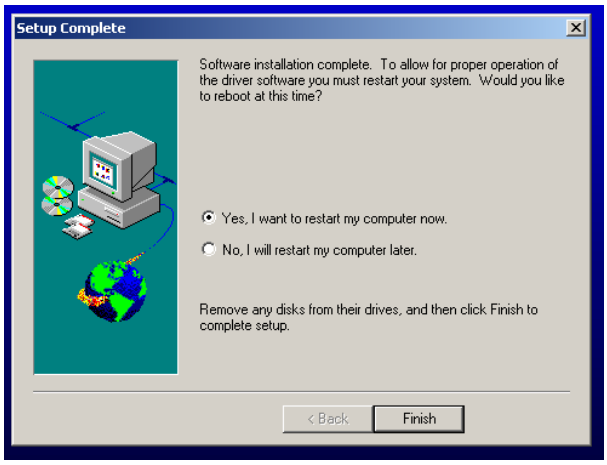

Figura 3.13: Reinicialización del SO

Una vez acabada la instalación del SR 5.1.1 reiniciamos el PC para que los cambios tengan efecto. A continuación instalamos el SR511SP1.zip ejecutando el archivo setup.exe.

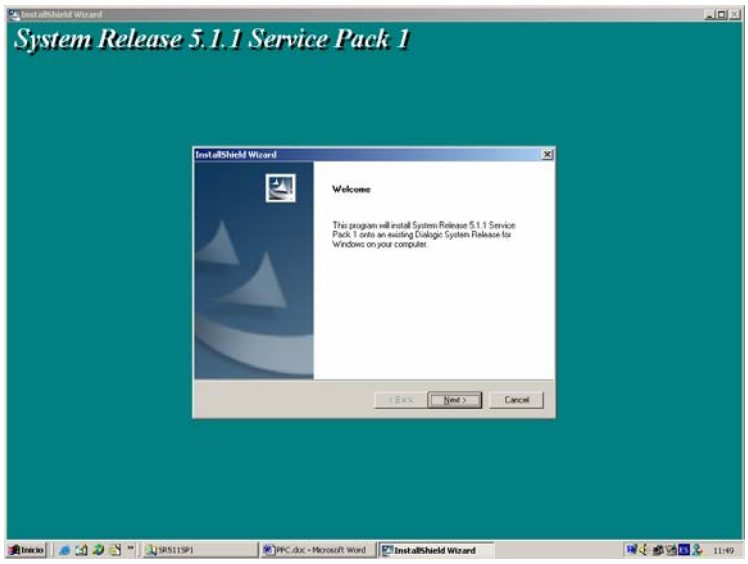

Figura 3.14: instalación del SR 5.1.1 SP 1 (I)

Una vez acabada la instalación del SR 5.1.1 SP 1reinciamos el PC para que los cambios tengan efecto. Al reiniciar vemos la carpeta que se ha instalado en el menú de inicio.

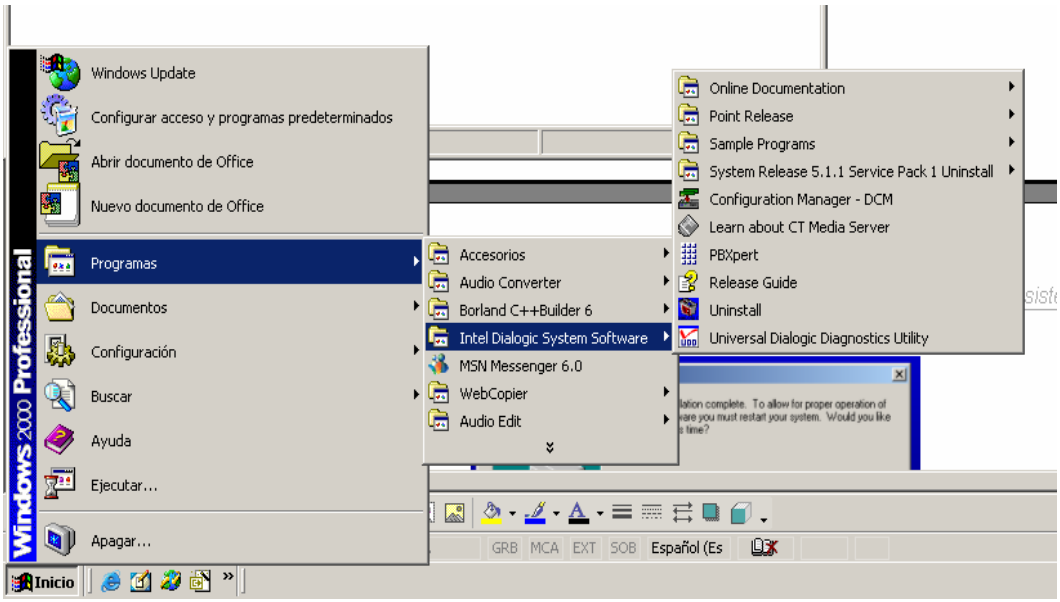

Figura 3.15: Carpeta del software de Dialogic creada con la instalación del SR 5.1.1 y el SP 1

Hay varias aplicaciones que nos proporcional Dialogic, como aplicaciones de diagnóstico (*Universal Dialogic Diagnostics Utility*), de configuración de los parámetros de la PBX que conectamos al sistema (*PBXpert*), de configuración del hardware instalado (*Configuration Manager - DCM*), de documentación y algunas aplicaciones de ejemplo. A continuación debemos configurar el hardware y activarlo, para ello utilizaremos el *Configuration Manager – DCM.*

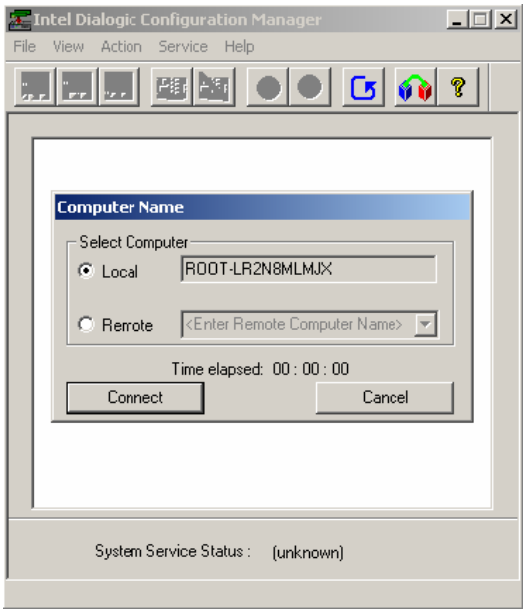

Figura 3.16: Iniciamos el DCM

Seleccionamos Local ya que vamos a configurar una tarjeta que esta en nuestro PC, si instalásemos SNMP podríamos manejar en remoto varias tarjetas instaladas en otros PC's.

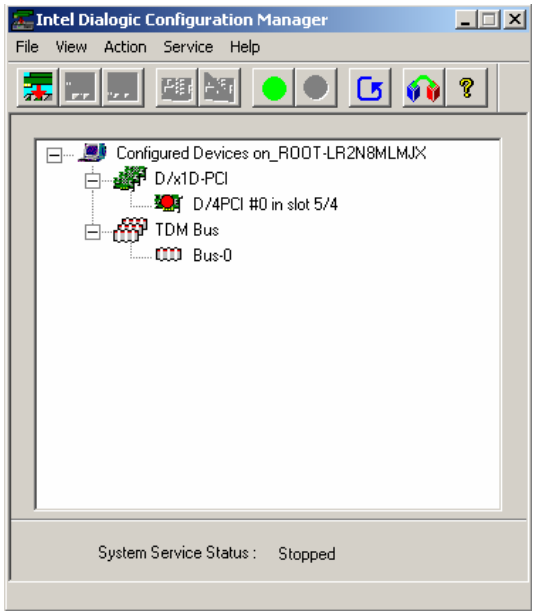

Figura 3.17: Hardware disponible y su estado actual

Una vez inicializado el DCM vemos el hardware que ha encontrado en nuestro sistema y su estado, vemos que tenemos nuestra tarjeta D/4PCI que el SO la ha inicializado como dispositivo 0 (este número de tarjeta no hace falta para abrir los puertos en la aplicación IVR) y que esta pinchada en el spot 5/4. El estado actual de la tarjeta es parada, y como el SW4 está en ON cualquier llamada a alguna de las extensiones de la tarjeta dará comunicando (OFFHOOK) hasta que inicialicemos la tarjeta desde el DCM, pero antes de inicializar la tarjeta configuraremos unas opciones. Para eso seleccionaremos la tarjeta con el ratón y pincharemos en el icono de configuración.

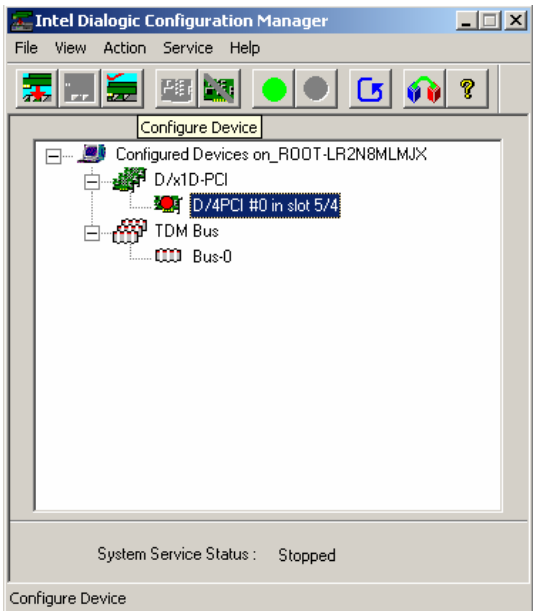

Figura 3.18: Configuración de la tarjeta (I)

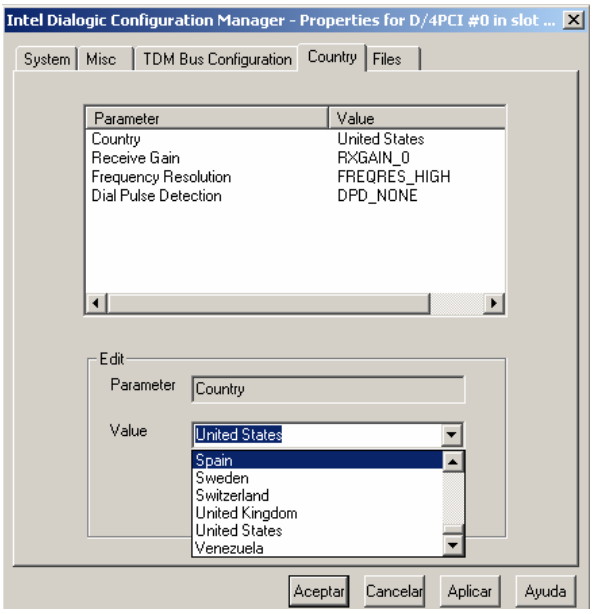

Figura 3.19: Configuración de la tarjeta (II)

Solo cambiaremos el país donde esta la tarjeta para el seleccionar el tipo de señalización analógica correcto.

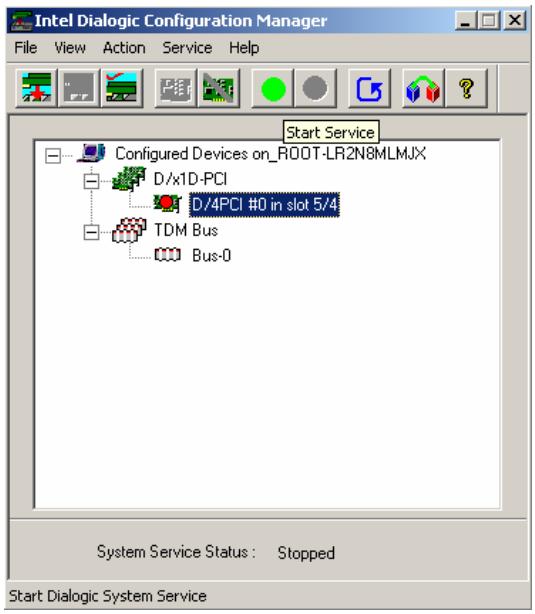

Figura 3.20: Inicialización de la tarjeta

Ahora inicializamos la tarjeta pulsando el icono de *start*, si no hay ningún problema nuestra tarjeta se inicializará y quedara en el esto de *running*. Ahora nuestra tarjeta ya esta preparada para ser utilizada por cualquier aplicación.

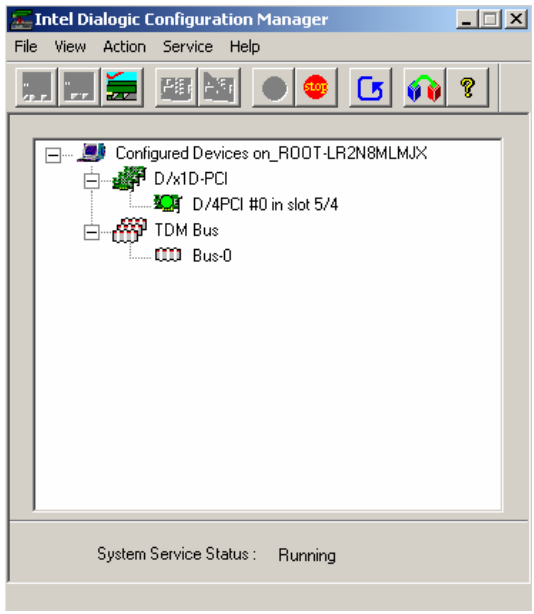

Figura 3.21: Tarjeta inicializada

El resto de configuraciones de la tarjeta que sean necesarias se realizan desde la aplicación que se utilice.

3.4 VerbioTTS

3.4.1 Prerrequisitos

Los productos *Verbio* actualmente están disponibles únicamente para sistemas operativos Windows en sus variantes 2000/XP, aunque actualmente se está en fase de pruebas internas de una versión para Linux. Antes de proceder a la instalación del software es conveniente verificar que el hardware utilizado reúne los requisitos mínimos indispensables para soportar la carga computacional y de memoria consumidos por los sistemas de síntesis del habla *Verbio*.

El uso de los sistemas de síntesis del habla y de las herramientas que constituyen la variedad de productos englobados bajo el nombre comercial *Verbio* requiere la instalación de un conjunto de módulos distribuidos en instaladores independientes. De este modo, únicamente es necesario descargar e instalar, en el orden adecuado, aquél o aquellos módulos estrictamente necesarios. El objetivo de este apartado es guiar a los instaladores para que puedan determinar los módulos que requieren, así como para que puedan proceder a su posterior instalación. Los módulos pueden clasificarse en base a distintos criterios, aunque las principales subdivisiones podrían ser:

• Módulos obligatorios y módulos opcionales
• Módulos de servidor y módulos de cliente
• Módulos de síntesis del habla

3.4.2 Módulos obligatorios

Los motores de síntesis del habla *Verbio* están diseñados para funcionar en una arquitectura cliente-servidor. Es decir, se requiere uno o más servicios (servidores) que atienden las peticiones de uno o más clientes. Entendemos por cliente toda aplicación que utilice recursos de síntesis del habla. En particular, todas aquellas máquinas que contienen el dispositivo de audio (tarjeta CTI o de sonido). Esta arquitectura permite diseñar un entorno de trabajo fácilmente escalable y con redundancia. En entornos de poca carga computacional o en aquellos que lo requieran, el cliente y el servidor pueden estar funcionando en la misma máquina.

Componentes del servidor (Verbio Server Engine)

Debe instalarse en todas aquellas máquinas destinadas a alojar un servidor (sólo se permite un único servidor por máquina). Para instalar Verbio Server Engine, debe ejecutarse el instalador **VerbioEngines.exe**. Este software hay que descargárselo de la web de Verbio **[12].**

Una vez seleccionado el directorio de instalación, aparece la ventana de selección de los componentes que deben instalarse. Si en la máquina actual únicamente debe instalarse el servidor, se seleccionará sólo la opción Componentes del servidor. En caso de que se desee instalar el cliente y el servidor en la misma máquina, como es nuestro caso deben seleccionarse los dos componentes.

Figura 3.22: Selección de componentes del Servidor y/o componentes del Cliente

Una vez instalados los componentes específicos de *Verbio*, el instalador deberá comprobar que en el equipo estén presentes los driver de las llaves de licenciamiento requeridas para la puesta en marcha del servicio. Las llaves son dispositivos hardware que deben conectarse al puerto paralelo o USB de la(s) máquina(s) en la(s) que se aloja(n) el(los) servidor(es). Para obtener las llaves debe contactar con su proveedor

habitual. En caso de que los driver no están instalados, se ofrece la posibilidad de hacerlo.

Figura 3.23: Opción de instalar los driver de las llaves de licenciamiento si no están previamente instalados.

Atención

En caso de que en el equipo haya insertada alguna llave, es imprescindible retirarla antes de proceder con la instalación de los driver. En caso contrario, podría perderse la información contenida en ellas.

Componentes del cliente (Verbio Client Engine)

Debe instalarse en todas aquellas máquinas destinadas a alojar uno o varios clientes, independientemente del tipo de transacción que deban realizar.

Para instalar Verbio Client Engine, debe ejecutarse el instalador **VerbioEngines.exe**. Una vez seleccionado el directorio de instalación, aparece la ventana de selección de los componentes que deben instalarse. Si en la máquina actual únicamente debe instalarse el cliente, se seleccionará sólo la opción Componentes del cliente. En caso de que se desee instalar el cliente y el servidor en la misma máquina, como es nuestro caso, deben seleccionarse los dos componentes.

3.4.3 Módulos opcionales

Componentes del servidor

Una vez instalados los módulos obligatorios del servidor según lo comentado en los apartados previos, deberá procederse a instalar aquellos componentes opcionales necesarios. *Verbio* distribuye módulos de síntesis del habla (locutores). Así, pues, para cada locutor que se desee utilizar, será necesario instalar su locutor correspondiente.

Locutores de síntesis

Para disponer de un locutor determinado en el servidor de síntesis, es imprescindible instalar su locutor correspondiente. En función de los idiomas que se deseen soportar, deben instalarse uno o más de los locutores disponibles para cada idioma. Mediante la función correspondiente del API de programación deberá seleccionarse el idioma (y, en ocasiones, el locutor) adecuado para cada caso, en función de las necesidades de la aplicación.

En la tabla siguiente se muestran los locutores disponibles en Verbio para cada uno de los idiomas soportados.

Idioma	Sexo	Edad	Nombre	Instalador
Catalán	Hombre	Adulto	Pau	Catalan-Pau
Catalán	Mujer	Adulta	Marta	Catalan-Marta
Español castellano	Hombre	Adulto	Carlos	Spanish-Carlos
Español castellano	Mujer	Adulta	Laura	Spanish-Laura
Español mexicano	Mujer	Adulta	Lupe	Mexican-Lupe
Euskera	Hombre	Adulto	Jon	Basque-Jon
Francés	Mujer	Adulta	Brigitte	en pruebas internas
Gallego	Hombre	Adulto	Freire	Galician-Freire
Inglés americano	Mujer	Adulta	Jane	en pruebas internas
Portugués	Mujer	Adulta	Adriana	Portuguese-Adriana

Tabla 3.1 : Locutores soportados por Verbio

Para este proyecto se ha contratado sólo una llave que soporta el idioma Español Castellano y se ha elegido, aunque se podría haber elegido cualquier locutor de este idioma, a la locutora Laura.

Componentes del cliente

Dentro de los módulos adicionales para el cliente, encapsulados dentro del instalador **VerbioDeveloper.exe [13]** se encuentran los siguientes componentes:

Figura 3.24: Conjunto de componentes disponibles para el equipo cliente

Plataformas de desarrollo de aplicaciones (Tipos de cliente)

Verbio proporciona un conjunto de plataformas de desarrollo para que integradores de múltiples entornos puedan incorporar tecnologías del habla a sus productos. Para cada una de las plataformas disponibles se proporciona:

- Software Development Kit (SDK): Conjunto de ficheros destinados a programar nuevas aplicaciones que incorporen reconocimiento y/o síntesis del habla (librerías, header, etc.).
- Ejemplos: Códigos de ejemplo de la utilización del SDK
- Documentación: Descripción de las funciones contenidas en el SDK (Function Referente).

La decisión del SDK a utilizar depende de un conjunto de factores que a continuación se detallan con la finalidad de que cada integrador encuentre aquél que mejor se adapte a su entorno de programación y a sus conocimimentos. La tabla siguiente es un resumen de las particularidades de cada SDK:

SDK	Entorno de programación	Hardware utilizado	Compatibilidad	Herramientas avanzadas	DLL contenedora
Advanced	Microsoft Visual Studio C/C++	Cualquier tarjeta CTI o dispositivo I/O	Incompatible con versiones anteriores	Sí	verbiolib.dll
Dialogic	Microsoft Visual	Tarjetas CTI	Compatible con	No	vxxxlib.dll

SDK	Entorno de programación	Hardware utilizado	Compatibilidad	Herramientas avanzadas	DLL contenedora
	Studio C/C++ Otros compiladores C/C++	Intel Dialogic	Verbio Dialogic (vx_)		
Library	Microsoft Visual Studio C/C++ Otros compiladores C/C++	Cualquier tarjeta CTI o dispositivo I/O	Compatible con Verbio Library (vox_)	No	voxlib.dll
CT ADE	Graphical VOS (Parity) - CT ADE	Tarjetas CTI Intel Dialogic	Compatible con versiones anteriores de IVR	No	ivr.dll

Tabla 3.2: Selección del SDK de interés

Es posible utilizar todos los SDK de desarrollo desde compiladores que no trabajen en C/C++, aunque la programación requerirá de un trabajo previo consistente en importar todas las funciones contenidas en la DLL del SDK de interés dentro del entorno de programación.

Verbio incorpora la plataforma de desarrollo Advanced para facilitar el uso de los motoros de reconocimiento y síntesis del habla, sobretodo de las nuevas funcionalidades incorporadas. El resto de plataformas se han actualizado para poder disfrutar también de las nuevas prestaciones de Verbio sin perder la compatibilidad con las versiones anteriores, aunque ello implique un uso algo más complejo que el de la nueva plataforma.

En nuestro caso hemos utilizado la plataforma Dialogic ya que estamos usando una tarjeta CTI de este tipo.

3.4.4 Requisitos hardware

A la hora de determinar el (los) equipo(s) necesario(s) para cada instalación, es conveniente tener en cuenta los aspectos que más influyen en el consumo de recursos hardware. Entre los aspectos más determinantes destacan los locutores utilizados y la distribución de la carga computacional.

Locutores utilizados

El uso de síntesis del habla consume una considerable cantidad de memoria RAM del sistema, de modo que ésta deberá adecuarse al tipo y cantidad de locutores utilizados. Verbio permite almacenar los datos del sintetizador básicamente todos en disco, de modo que se reduce la cantidad de memoria necesaria, aunque a costa de un tiempo de respuesta ligeramente superior (sobretodo en entornos sobrecargados). Por lo tanto, únicamente se aconseja utilizar esta estrategia en aquellos entornos que requieran pocas transacciones de síntesis simultáneas (del orden de 5) y que tengan limitaciones insalvables de memoria. El consumo de memoria debido a la activación de locutores es elevado, aunque permanece prácticamente constante durante todo el periodo de utilización, independientemente de las peticiones que se le formulen. De este modo, la memoria necesaria viene condicionada por la cantidad de locutores a utilizar y no por la cantidad de peticiones simultáneas que deba ser capaz de procesar el sintetizador. Como punto de partida, para un uso estándar de 2 locutores, es aconsejable un mínimo de 512 MBytes de memoria RAM (preferiblemente de acceso rápido tipo DDRAM). Respecto a la carga computacional, se pueden tener en cuenta las siguientes mediciones:

Prestaciones máquina	Peticiones simultáneas
Intel Pentium III a 1 GHz	17
Intel Pentium IV a 2,4 GHz	27

Tabla 3.3: Peticiones simultáneas en función del tamaño del vocabulario

3.4.5 Política de licencias

Verbio dispone de licencias diferenciadas para reconocimiento y para síntesis, nos centraremos en las de síntesis que son las únicas que nos interesan.

Dentro de las licencias de síntesis hay licencias distintas para cada uno de los idiomas soportados. De este modo, sólo es necesario adquirir las licencias en cada caso, pudiéndose actualizar o incrementar el número de las mismas a medida que se incrementa las necesidades del sistema.

Licencias de síntesis

Las licencias de síntesis del habla se dividen en dos grandes grupos: licencia de motor y licencia de idioma. Ambas licencias se consumen durante el proceso de síntesis; es decir, mientras el motor de síntesis del habla genera y manda las muestras al cliente. Para que el proceso de síntesis se lleve a cabo, es necesario disponer de una licencia de motor y de una licencia de cada uno de los idiomas que estén presentes en el texto a sintetizar. En caso de que falte alguna de ellas, no se sintetizará ningún fragmento del texto.

Soporte de licencias

Las licencias pueden facilitarse en formato software (fichero) o hardware: mochila (sentinel) de puerto paralelo o USB. Para la realización de este proyecto se ha adquirido en formato hardware, mochila Rainbow Sentinel Superpro en su versión puerto USB, ya que presenta la ventaja de ser reutilizable en cualquier máquina.

Figura 3.25: Fotografía mochila Rainbow Sentinel Superpro

3.4.6 Puesta en marcha del servidor Verbio

Inicio de Verbio Server Configuration Manager

Una vez instalados los módulos obligatorios del servidor y aquellos módulos opcionales necesarios, puede procederse a la puesta en marcha del servidor síntesis de habla Verbio. Para ello debe usarse Verbio Server Configuration Manager.

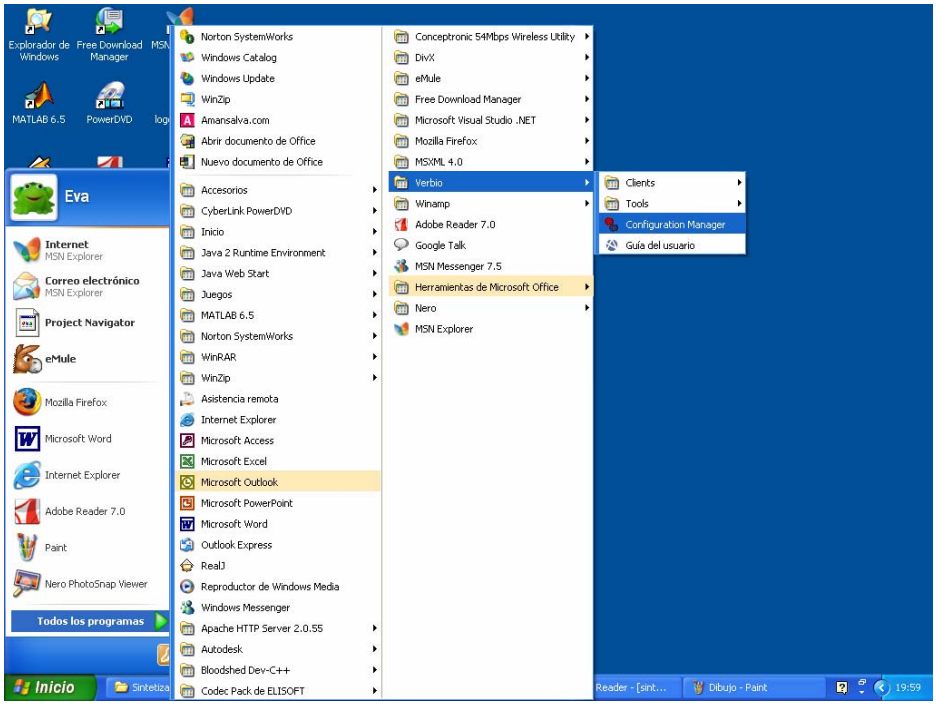

Figura 3.26: Arranque del configurador de Verbio, Verbio Server Configuration Manager

Las operaciones que pueden realizarse desde Verbio Server Configuration Manager son las siguientes:

- Arrancar y parar el servidor de síntesis del habla Verbio
- Especificar los locutores que se utilizan, así como el locutor por defecto para cada idioma. El locutor por defecto es el que se utilizará en el proceso de síntesis

siempre y cuando no se haya especificado ningún otro locutor y/o idioma previamente.

- Incrementar el número de licencias disponibles
- Determinar si el servicio de síntesis del habla de Verbio debe arrancar automáticamente al iniciarse el equipo y la frecuencia de trabajo del motor de síntesis.

Asimismo, Verbio Server Configuration Manager permite visualizar la siguiente información:

- Las configuraciones instaladas en el equipo, así como el número de licencias disponibles para cada uno de ellos y el tamaño que ocupan en caso de ser activados.
- Las licencias disponibles para motores de reconocimiento, síntesis y utilización SAPI.
- El número de serie del servidor, la versión, así como el puerto por el que opera. También se muestra el número de serie de la llave (sentinel) que contiene las licencias si está insertada en la ranura. En caso contrario, el valor es -1.

Con Verbio Server Configuration Manager se pueden realizar otras operaciones relacionadas con el reconocimiento de voz, pero estas no son relevantes para nuestro proyecto ya que nuestra aplicación no va a disponer de esa funcionalidad, ésta se propondrá más adelante como una línea futura a la aplicación desarrollada. Nosotros nos centraremos en la opción de síntesis o TTS.

Una vez arrancado Verbio Server Configuration Manager, el primer paso consiste en verificar que se dispone de licencias para todas aquellos locutores (pestaña TTS speaker) que desean utilizarse. Si no conectamos la mochila que lleva las licencias nos encontraremos lo siguiente, que muestra la información del locutor usado pero que no dispone de ninguna licencia.

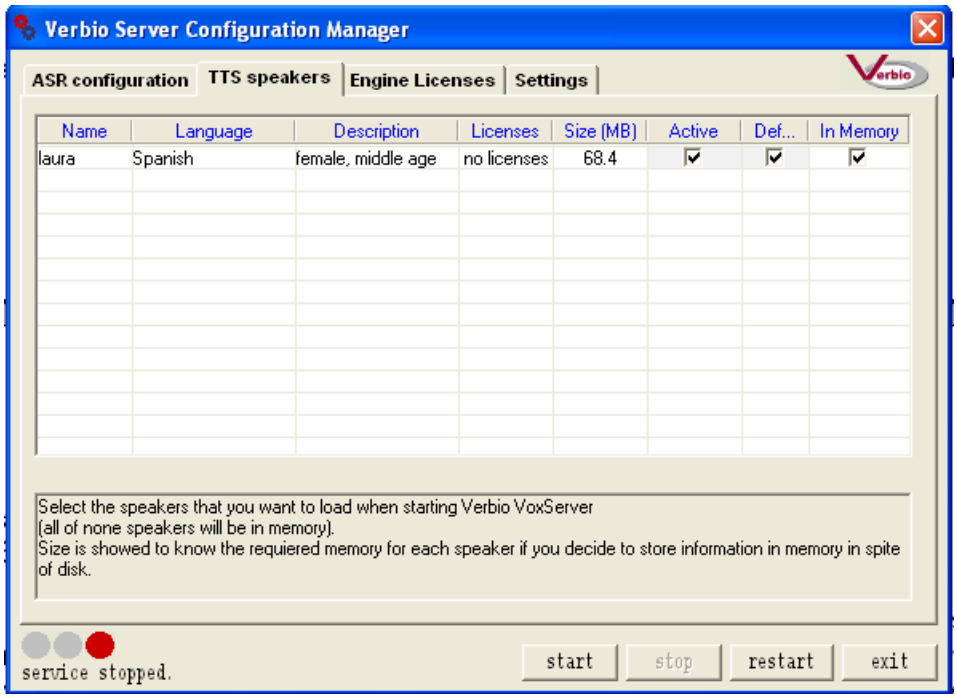

Figura 3.27: Información de TTS speakers

Este problema se resuelve conectando la mochila de licencias, que deberá estar siempre conectada al servido (en este caso también cliente) de la aplicación.

Especificación de locutores de interés

Tras haber verificado que se disponen de licencias tanto de motor como de los locutores deseados, es necesario indicar al servidor qué locutores debe cargar, así como los locutores por defecto para cada idioma. En nuestro caso es sencillo ya que sólo vamos a disponer de un locutor.

Debe seleccionarse la casilla de la columna Active de todos los locutores que deban estar disponibles en el servidor una vez esté operativo.

Si tuviésemos más de un idioma de trabajo, para cada uno de ellos se debería seleccionar un locutor por defecto, que será el utilizado en aquellas peticiones en las que no se haya indicado específicamente otro locutor para el idioma afectado. Por defecto se

selecciona la casilla Default del primer locutor activado para cada idioma, aunque puede modificarse esta selección posteriormente.

Finalmente, puede optarse por acceder a disco o a memoria (esta medida afecta a todo el conjunto de locutores seleccionados) durante los procesos de síntesis. Si se desea acceder a memoria (mayor velocidad aunque mayor consumo de memoria) durante el proceso de síntesis debe seleccionarse la casilla In Memory de cualquiera de los locutores activados.

También es posible indicar la frecuencia de trabajo del servidor de síntesis. La mayoría de los locutores de Verbio permiten trabajar a 8 KHz (para aplicaciones telefónicas) y a 16 KHz (para aplicaciones desktop). Podrá seleccionarse la deseante mediante el desplegable situado en la pestaña Settings. Nosotros seleccionaremos 8 KHz ya que estamos trabajando en una aplicación es telefónica.

Atención

Es imprescindible detener y volver a arrancar el servicio para que los cambios en la configuración tengan efecto mediante el botón Restart.

Después de configurar el Verbio Server Configuration Manager nos queda de la siguiente manera:

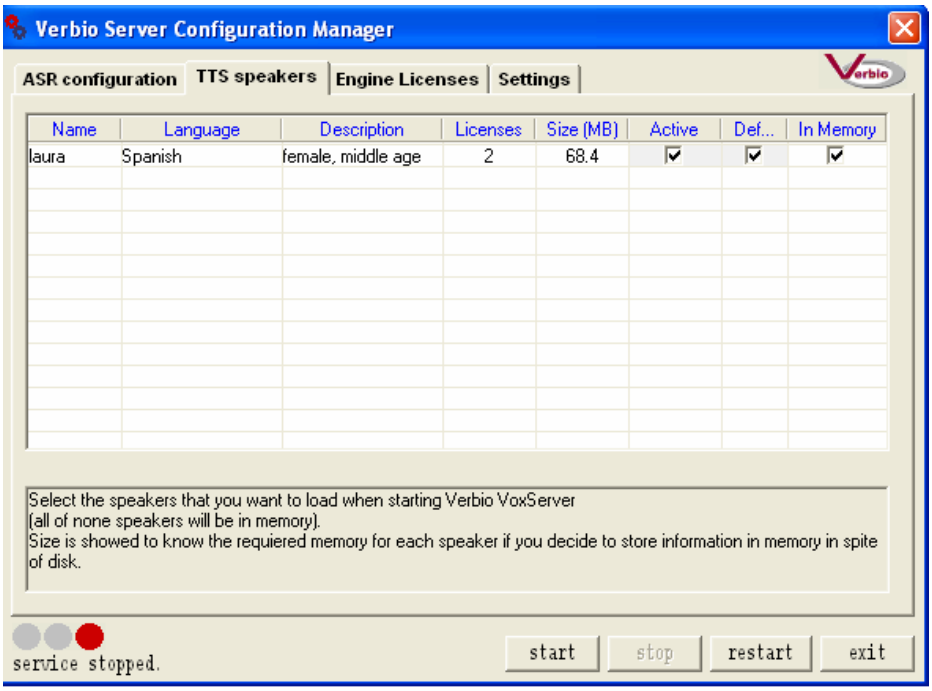

Figura 3.28: Especificación y configuración de los locutores de síntesis

El uso de los sistemas de síntesis del habla de Verbio requiere, en función de cada caso, la adquisición de varios tipos de licencias, de modo que se adopten lo mejor posible al uso que se vaya a hacer de ellos.

- Licencias de motor TTS: Es necesario disponer de tantas licencias de motor de síntesis como peticiones simultáneas de síntesis deseen poder atenderse en el servidor, independientemente de los idiomas utilizados. La cantidad de licencias de síntesis disponibles se muestran en la pestaña Engine Licenses.
- Licencias de acceso vía Sapi: Es necesario disponer de tantas licencias de motor de acceso vía Sapi como peticiones simultáneas deseen poder realizarse desde una aplicación compatible SAPI hacia los locutores de Verbio utilizando este protocolo. La cantidad de licencias de acceso SAPI disponibles se muestran en la pestaña Engine Licenses.
- Licencias específicas para cada idioma (síntesis): En función de los idiomas que se deseen utilizar en síntesis es necesario disponer de las suficientes licencias (tantas como peticiones simultáneas se deban procesar) de dichos idiomas (la licencia para un idioma permite utilizar cualquier locutor de ese idioma).

Lógicamente, el número de licencias de un idioma no debe ser superior al número de licencias de motor TTS disponibles (estarían infrautilizadas). La cantidad de licencias disponibles para cada locutor se muestra en la pestaña TTS Speakers.

Para nuestra aplicación, como ya se ha comentado, se han contratado licencias para un solo idioma, el castellano y para dos peticiones simultáneas. Como se ha expuesto anteriormente, no hay ningún problema a la hora de ampliar, en un futuro, el número de locutores y/o el de peticiones simultáneas. Bastaría con adquirir las licencias adecuadas.

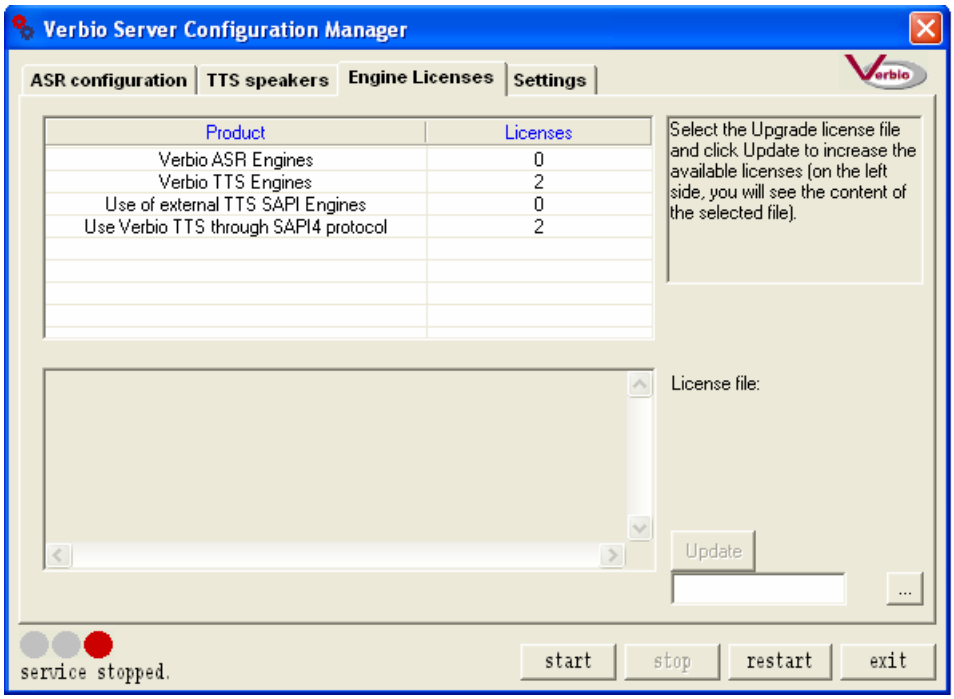

Figura 3.29: Consulta y actualización de las licencias disponibles

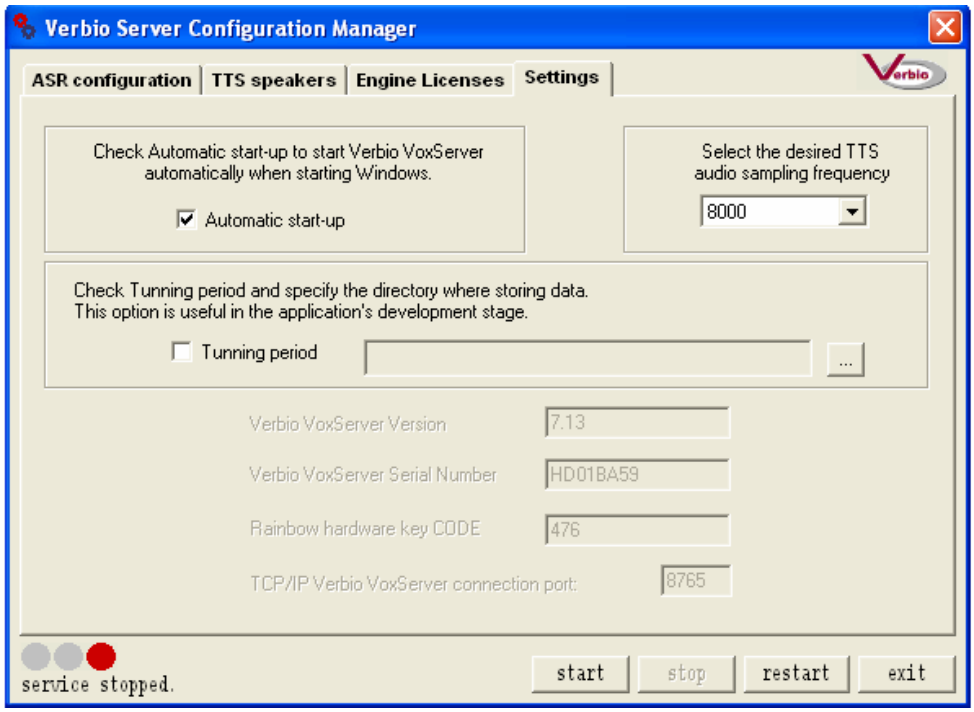

Figura 3.30: Números de serie del producto y de la llave necesarios para la obtención de licencias

Una vez seleccionadas las configuraciones y/o locutores de interés, debe arrancarse el servidor de síntesis pulsando el botón Start. En caso de disponer licencias para todos los locutores seleccionados, Verbio se instalará como un servicio de Windows, arrancando conjuntamente con el sistema operativo si se ha seleccionado la opción Automatic Start-up de la pestaña Settings. En caso contrario deberá arrancarse manualmente mediante Verbio Server Configuration Manager o mediante el panel de control de servicios de Windows.

3.4.7 Verbio Read Aloud

Para comprobar que hemos hecho la instalación correctamente, Verbio dispone de una herramienta destinada a probar los locutores disponibles mediante la especificación de cualquier texto y la manipulación de sus parámetros de funcionamiento básicos.

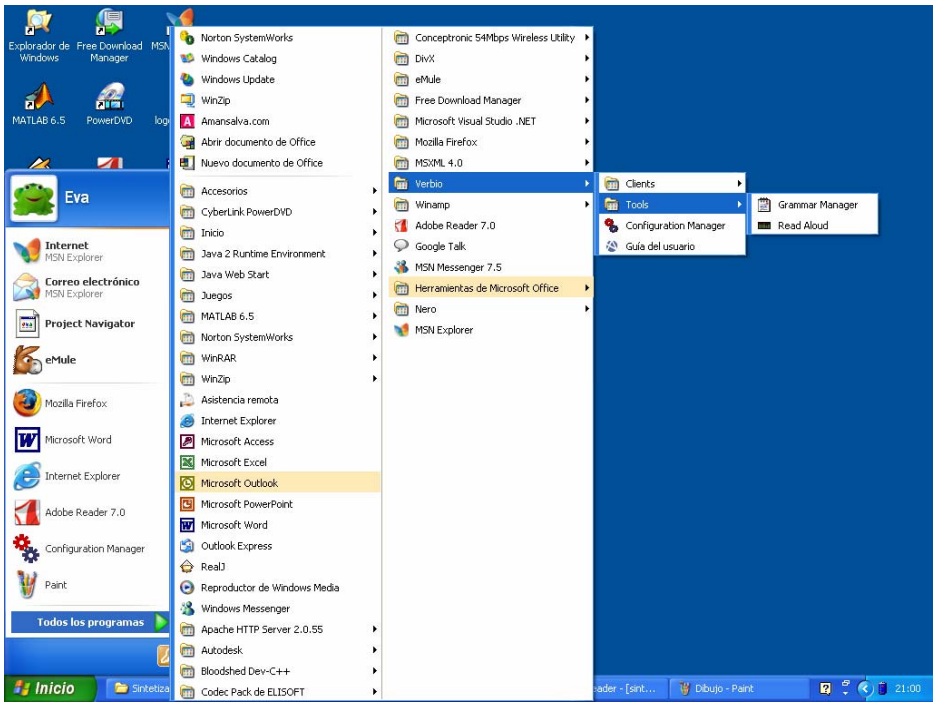

Figura 3.31: Arranque de la herramienta Verbio Read Aloud

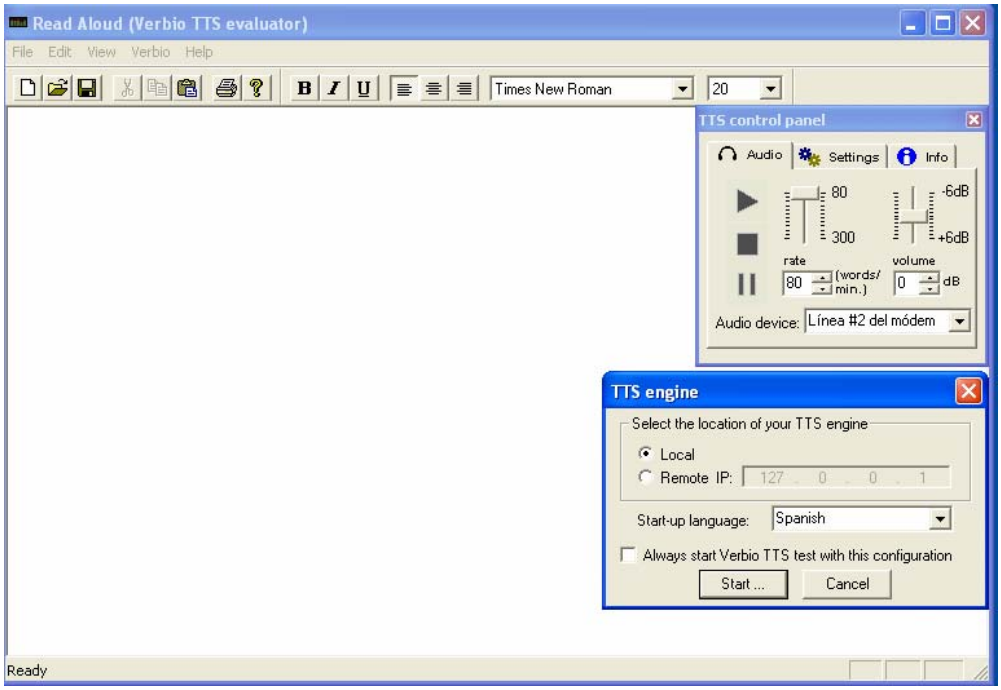

Figura 3.32: Entorno Verbio Read Aloud

Como servidor y cliente están en la misma máquina, en la ventana TTS engine marcamos local y pinchamos Start. Si no ha ocurrido ningún error, escribimos un texto

en la consola y pinchamos en el botón de reproducir de la ventana TTS control panel, si escuchamos la reproducción del texto que hemos introducido es que funciona correctamente, si obtenemos algún error habría que revisar la instalación desde el principio.

3.5 Bases de datos

3.5.1 Introducción

En este apartado se describe cual es la base de datos utilizada en el sistema y su estructura con el fin de capacitar al administrador del sistema para que sea capaz de instalar y mantener la base de datos utilizada.

El sistema de base de datos elegida para el desarrollo del proyecto es Microsoft Access. Posteriormente, a la hora de poner en marcha esta aplicación, habrá que redireccionar varias tablas de la base de datos en Acces a la base de datos de Oracle que usa la UPCT. Para la aplicación hemos utilizado dos bases de datos, una que contiene los datos de configuración de sistema y logs, y otra que contiene los datos de las asignaturas, notas, usuarios, y estructura de flujo del programa para una modificación básica sin necesidad de reescribir código ni recompilar.

3.5.2 Instalación y configuración

La aplicación accede a la base de datos para realizar consultas, modificaciones, añadir datos o tablas, etc. mediante SQL *"Structured Query Language"* , esto es un lenguaje de bases de datos que pretende ser un lenguaje común para las bases de datos relacionales mediante sentencias de cadenas de caracteres.

Para que la aplicación pueda acceder a los datos de la base de datos debemos configura una conexión ODBC. Este conector se define mediante la utilizad de "Orígenes de datos (ODBC)" que proporcionan los sistemas operativos XP, NT y Windows 2000. Se debe crear un DSN *Data Source Name*. Al ejecutarlo se nos abre una ventana con varias pestañas.

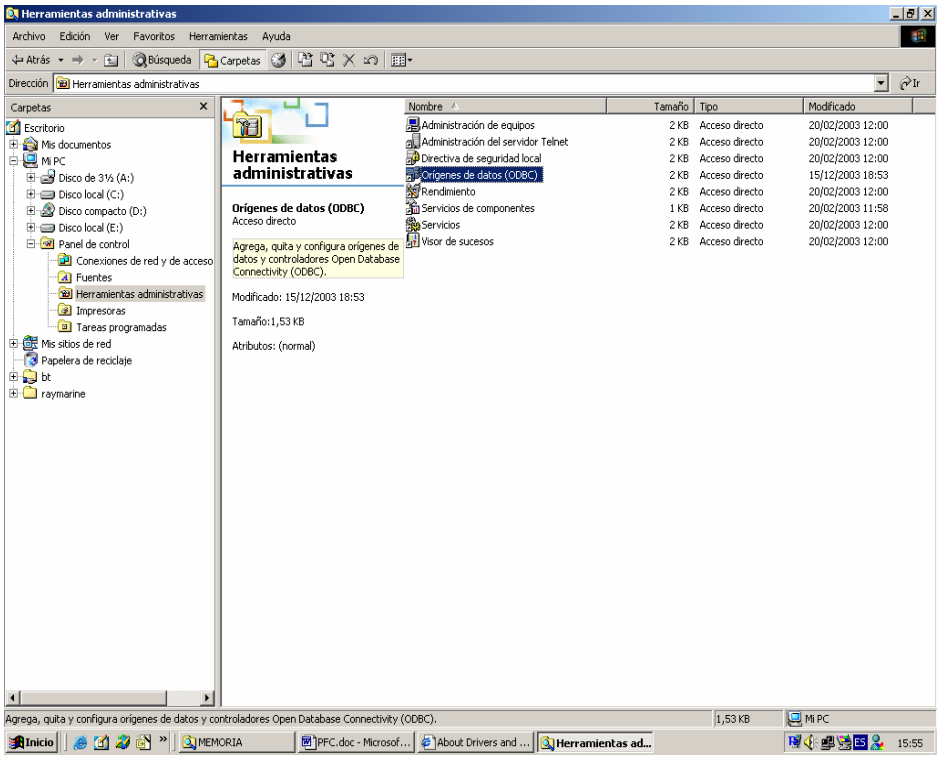

Figura 3.33: Panel de control-orígenes de datos (ODBC)

De las siete pestañas que tenemos nos interesan solo dos, DSN de usuario y DSN de sistema, estas son para crear y gestionar DSN de bases de datos que son usuario o de sistema, esto es que solo pueden ser utilizadas por el usuario que las crea o que pueden ser de sistema y pueden ser configurados los privilegios para el acceso a ellas. Al crear un DSN lo que hacemos es crear un conector con el *driver* de la base de datos a la que vamos a acceder y le asignamos un nombre que nos servirá de alias para que la función de conexión a la base de datos acceda por ese alias realizando una petición al SO.

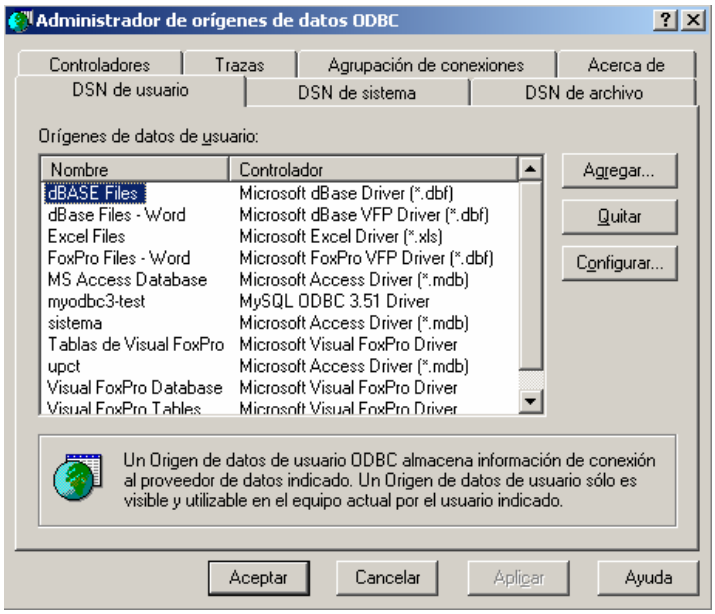

Figura 3.34: Administrador de origen de datos ODBC

Nuestra aplicación maneja dos bases de datos, una con el alias de *sistema* que contiene la configuración de la tarjeta y aplicación y otra cuyo nombre se determina en un campo de la tabla de configuración de la base de datos de *sistema*. Una vez que tenemos los dos ficheros con las bases de datos en su directorio, por ejemplo *ruta_de_la_aplicación/db/sistema.mdb* y *ruta_de_la_aplicación/db/upct.mdb* creamos el DSN para cada base de datos.

Utilizaremos DSN de usuario o de sistema dependiendo de si el PC en el que la aplicación va a correr es un servidor que siempre esta en un usuario determinado (administrador) o si la aplicación va a correr en distintos usuarios. Pinchamos en el botón agregar del administrador de origen de datos ODBC y nos da a seleccionar el tipo de origen de datos (controlador), seleccionamos *Microsoft Access driver (*.mdb).*

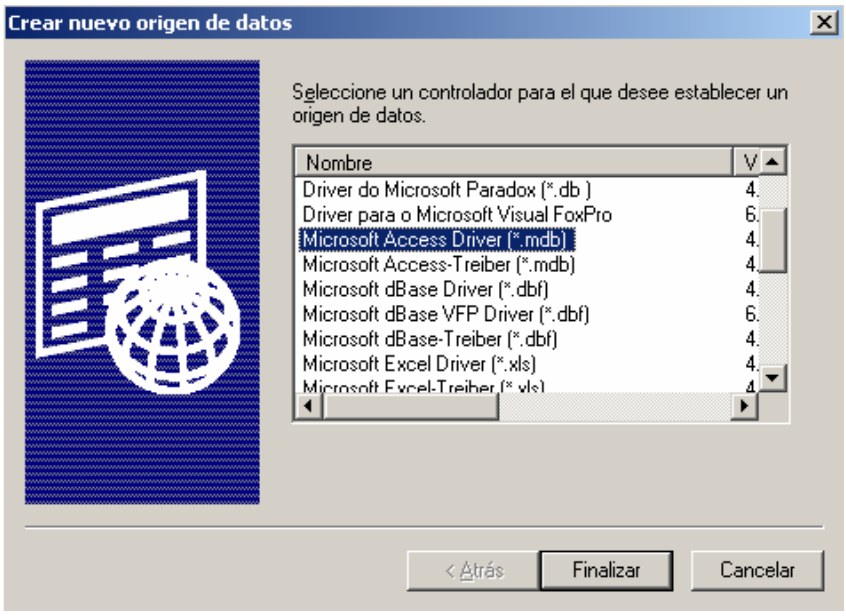

Figura 3.35: Selección del controlador de origen de datos

A continuación configuramos el controlador de origen de datos, introducimos el nombre del DSN, alias, en nombre del origen de datos y pinchamos en el botón seleccionar del marco base de datos, una ventana de selección de fichero se abre y seleccionamos el fichero de la base de datos de sistema, aceptamos y ya tenemos el DSN para la base de datos de sistema. Repetimos el proceso con la base de datos upct.mdb y la llamamos upct, este nombre de DSN habrá que especificarlo en la tabla de configuración de base de datos en la tabla de *config* de sistema.

Ya tenemos las bases de datos de la aplicación preparadas para su utilización.

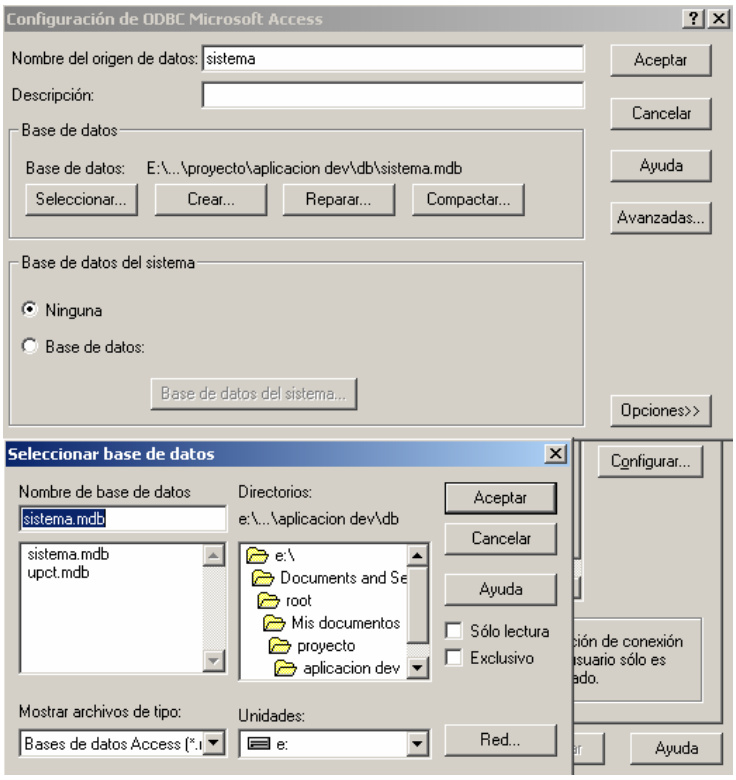

Figura 3.36: configuración del controlador de origen de datos

3.5.3 Estructura

Ahora entraremos en detalle en la estructura de las dos bases de datos y su funcionamiento. Empezaremos por la base de datos de sistema, la cual tiene un DSN llamado sistema y su origen de datos es sistema.mdb, esta base de datos debe de respetarse tanto sus tablas originales como su nombre, si se desea crear una nueva es aconsejable copiarla y borrar el contenido de sus tabla en lugar de ir creándola desde cero.

Base de datos de sistema

Esta base de datos contiene los parámetros de configuración de la tarjeta, de sus canales individuales, el nombre de la base de datos donde están los datos para realizar las consultas de notas etc. y el contenido histórico de mensajes de estado, logs, para comprobar el funcionamiento de la tarjeta y facilitar la resolución de fallos en caso de que haya algún error en ejecución.

Esta base de datos contiene dos tablas no relacionadas:

- Config, contiene los datos de configuración.
- Log, contiene los mensajes de sistema en tiempo de ejecución.

Tablas

Config	
Nombre del campo	**Tipo de datos**
Parámetro	Texto
Valor	Texto

Tabla 3.4: Campos de la tabla *config*

Tabla *config*; en esta tabla se almacenan pares parámetro-valor que contienen información para configurar la tarjeta y aplicación realizando consultas desde la aplicación a esta base de datos para leer los valores actuales de los parámetros, esto permite al administrador del sistema modificar parámetros que se configuran en tiempo de ejecución si necesidad de recompilar la aplicación, esta tabla tiene dos campos:

- *Parámetro*, campo de texto que contiene un string que identifica el parámetro.
- *Valor*, campo de texto que contiene un string con el valor del parámetro especificado en el campo *parametro*.

Los parámetros creados en la aplicación son:

Config	
Parámetro	**valor**
Alcatelbasic4/8transfer	R,
canaldb1	upct
canaldb2	
canaldb3	
canaldb4	
Maxrings	0
multidb	false

Config	
Parámetro	**valor**
pbx	upct
tiempoflash	10
upcttransfer	R,

Tabla 3.5: Parámetros utilizados en la aplicación

A continuación explicamos cada parámetro:

- *Canaldbx*: nos indica el DSN de la base de datos del canal x para realizar las consultas de datos, si los cuatro canales funciona con la misma base de datos con especificar el *canaldb1* es suficiente.
- *Maxrings*: indica el número máximo de tonos de llamada antes de que el canal atienda la llamada, si las extensiones implementan el protocolo de identificación de llamada este parámetro debe se puesto como mínimo a 2 para que la tarjeta sea capaz de recibir el número del llamante ya que este es transmitido entre el primer y el segundo tono de llamada.
- *Multidb*: con este parámetro indicamos si habrá una base de datos para cada canal o una para todos, escribiendo true o false respectivamente.
- *Pbx*: con este parámetro indicamos el nombre o marca de la central a la que esta conectado el sistema.
- *Tiempoflash*: con este parámetro indicamos la duración en milisegundos de duración del flash, pulsación R equivalente a pulsar uno en marcación decadita, necesario en muchos casos para pasar a modo de facilidades o datos en las centrales digitales como la transferencia de llamada.
- *PBXtransfer*: con este parámetro indicamos la secuencia para realizar transferencias de llamadas en la central telefónica a la que esta conectado el sistema, el parámetro tiene el nombre de la PBX a la que

esta conectado el sistema concatenado con la palabra transfer, en nuestro caso *upcttransfer*, para la alcatel Basic 4/8 seria *Alcatelbasic4/8transfer*. Los posibles caracteres son los números del 0 al 9, la R para marcación de flash y la coma para realizar una pausa.

Log	
Nombre del campo	**Tipo de datos**
Fecha	Texto
Texto	Texto

Tabla 3.6: Campos de la tabla *log*

Tabla *log*; en esta tabla se almacenan los mensajes de sistema mediante inserción de registros para así poder tener un histórico del funcionamiento de la aplicación y poder localizar y prevenir fallos en tiempo de ejecución, contiene dos campos:

- *Fecha*, contiene la fecha y hora a la que se produce el evento indicado en el campo texto.
- *Texto*, contiene un mensaje de la aplicación para indicar los eventos importantes como los errores en tiempo de ejecución.

Base de datos de UPCT

Esta base de datos contiene los datos necesarios para que la aplicación pueda proporcionar los servicios que tiene objeto, será esta base de datos la que irá relacionada con la base de datos Agora de Gestión Académica de la UPCT. Para ello contiene los datos de alumnos, asignaturas, consultas realizadas, convocatoria actual, claves de alumnos y datos de la estructura del diagrama de flujo de la aplicación para su modificación sin necesidad de cambiar código ni recompilar.

Estos datos se guardan en las siguientes tablas relacionadas:

- *Alumno*, contiene los datos del alumno.
- *Asignatura*, contiene los datos de la asignatura.
- *Notas*, contiene las notas de los alumnos de la asignatura.
- *Consultas*, contiene información de las consultas que se han realizado como la fecha y quien la realiza.
- *Estado*, contiene los estados de la aplicación con los datos necesarios para que cada estado realice su acción concreta.
- *Hilo*, contiene los saltos de hilo de ejecución permitidos en los menús de selección.
- *Clavesalumnos*, contiene las claves de los alumnos.
- *Convocatoria*, contiene los datos de la convocatoria actual.
- *Administrador*, contiene el/los administrador/es que pueden ver las consultas realizadas.

Tablas

Alumno	
Nombre del campo	**Tipo de datos**
Dni	Texto
Nombre	Texto
Apellidos	Texto
FechaNac	Número

Tabla 3.7: Campos de la tabla *alumno*

Tabla *alumno*: en esta tabla se almacenan los datos de los alumnos para permitirles el acceso a la base de datos mediante su DNI como código de usuario y una clave personal, que en un principio será su fecha de nacimiento contiene los siguientes campos:

- *DNI*, es la clave principal de la tabla, contiene el DNI del alumno.
- *Nombre*, contiene el nombre del alumno.
- *Apellidos*, contiene los apellidos del alumno.
- *FechaNac*, contiene la fecha de nacimiento del alumno.

Asignatura	
Nombre del campo	**Tipo de datos**
Codigo	Número
Nombre	Texto

Tabla 3.8: Campos de la tabla *asignatura*

Tabla *asignatura*: en esta tabla se almacenan las asignaturas, contiene los siguientes campos:

- *Código,* es la calve principal de esta tabla, contiene el código que identifica la asignatura.
- *Nombre*, contiene el nombre de la asignatura.

Notas	
Nombre del campo	**Tipo de datos**
DNI	Texto
Codigo	Número
Nota	Texto
Notanum	Número

Tabla 3.9: Campos de la tabla *notas*

Tabla *notas:* en esta tabla se almacenan las notas de los alumnos, contiene los siguientes campos:

- *DNI*, es la clave principal de esta tabla, contiene el DNI del alumno que tiene la nota.
- *Código*, contiene el código de la asignatura de la cual el alumno ha obtenido nota.
- *Nota*, calificación alfanumérica obtenida. Esta puede ser: Matricula de honor, Sobresaliente, Notable, Aprobado, Suspenso.
- *Notanum*, calificación numérica obtenida.

Consultas	
Nombre del campo	**Tipo de datos**
Fecha	Texto
DNI	Número
Asignatura	Número
Nota	Número

Tabla 3.10: Campos de la tabla *consultas*

Tabla *consultas:* esta tabla contiene registros de las consultas que los alumnos van realizando, guardando información a cerca de la consulta realizada, contiene los siguientes campos:

- *Fecha*, es un campo de texto donde se escribe la fecha y hora de la consulta.
- *DNI*, es el DNI que indica el usuario sobre el que se hace la consulta.
- *Asignatura*, es el código de la asignatura sobre la que se realiza la consulta.
- *Nota*, contiene la nota numérica obtenida por el usuario con ese DNI en la asignatura *código*, esta nota se almacena en la consulta realizada ya que después de una revisión la nota puede variar.

Clavesalumnos	
Nombre del campo	**Tipo de datos**
DNI	Número
Clave	Número

Tabla 3.11: Campos de la tabla *clavesalumnos*

Tabla *clavesalumnos:* esta tabla contiene las claves de acceso de los alumnos, contiene los siguientes campos:

- *DNI*, el DNI del alumno que contiene la clave.
- *Clave*, clave de acceso al sistema. Inicialmente este campo estará compuesto por las fechas de nacimiento de los alumnos, y se irá modificando conforme el alumno modifique su clave.

Convocatoria	
Nombre del campo	**Tipo de datos**
Año	Texto
Conv	Texto

Tabla 3.12: Campos de la tabla convocatoria

Tabla *convocatoria:* en esta tabla se refleja la convocatoria para la cual el sistema tiene notas, contiene los siguientes campos:

- *Año*, se corresponde con el curso actual.
- *Conv*, este campo puede valer “F”, si la convocatoria actual es Febrero, “J”, si es Junio, “S”, Septiembre, ó “D” si la convocatoria a la que va a hacer referencia es Diciembre.

Estado	
Nombre del campo	**Tipo de datos**
Actual	Número
Error	Número
Siguiente	Número
Hilo	Número
Play	Texto
Playerror	Texto
Rec	Texto
Maxdtmf	Número
Maxtime	Número
Seleccion	Número
Hilosiguiente	Número
Descripción	Texto

Tabla 3.13: Campos de la tabla *estado*

Tabla *estado:* esta tabla contiene los estados del flujo de ejecución del programa con los datos de configuración y necesarios para que cada estado realice la acción que debe, después de la explicación de los campos de cada tabla se explicara el funcionamiento de las tablas estado e hilo que son las que definen el comportamiento de ejecución de la aplicación, contiene los siguientes campos:

- *Actual*, este campo junto con el campo hilo definen la clave principal de esta tabla, este campo determina el número secuencial del hilo de ejecución en el que la aplicación esta, simplemente indica una posición dentro de una secuencia.
- *Error*, este campo determina el número secuencial del hilo de ejecución actual al que saltara en caso de que la función del estado actual genere un error.
- *Siguiente*, este campo determina el número secuencial del hilo de ejecución actual en el que debe seguir la aplicación en caso de

ejecutarse la función del estado actual en caso de ejecutarse con normalidad.

- *Hilo*, este campo determina el hilo de ejecución al que pertenece el estado que se indica el campo *actual*, un hilo representa un segmento dentro de un árbol, más adelante se explicaran estos conceptos dentro de la aplicación para entender el esquema de estados e hilos.
- *Play*, este campo indica el texto que se reproduce en el estado actual .
- *Playerror*, este campo indica el texto que se reproduce en caso de que se encuentre alguna incidencia pero dentro de la ejecución normal del estado, si se produce un salto de estado a *error* no se reproduce este texto.
- *Rec*, este campo indica un mensaje del usuario en el estado actual , este campo no se utiliza en la aplicación pero se ha incluido en la base de datos para una posible ampliación futura de la aplicación.
- *Maxdtmf*, este campo indica el número máximo de dígitos que se esperan en este estado.
- *Maxtime*, este campo indica el tiempo máximo, en segundos, para que se introduzcan los dígitos esperados.
- *Selección*, este campo indica la función que se debe realizar en este estado, esta funciones don definidas en el apéndice A de la aplicación.
- *Hilosiguiente*, este campo indica el hilo al que se cambia en este estado para seguir un nuevo flujo de programa.
- *Descripción*, este campo explica la función que se realiza en este estado.

Hilo	
Nombre del campo	**Tipo de datos**
Hilo	Número
Dtmf	Número
Hilosiguiente	Número

Tabla 3.14: Campos de la tabla *hilo*

Tabla *hilo:* esta tabla contiene los saltos de hilo que se realizan en los estados de menús correspondientes a cada digito DTMF pulsado, contiene los siguientes campos:

- *Hilo*, junto con dtmf es la clave principal, este campo indica el hilo al que pertenece el menú de selección para buscar cual es el hilo siguiente a donde saltar en función del digito marcado.
- *Dtmf*, este campo contiene los dígitos que se pueden marcar como selección en el menú en el que se encuentra.
- *Hilosiguiente*, este campo contiene el hilo al que se salta en función del digito dtmf pulsado en el menú de selección.

Administrador	
Nombre del campo	**Tipo de datos**
Login	Texto
Password	Texto

Tabla 3.15: Campos de la tabla *administrador*

Tabla *administrador:* esta tabla contiene el login y password del administrador/es de la aplicación:

- Login, login del administrador con el cual debe autentificarse.
- Password, contraseña del mismo. Esta clave la puede modificar desde el menú de administrador.

Una vez visto la estructura de cada tabla, de ambas bases de datos, y explicado el contenido de cada campo vemos que la aplicación interactuará con las bases de datos no solo para realizar consultas sino también en el proceso de arranque de la aplicación para consultar los datos que necesita para configurar tanto algunos parámetros de la tarjeta, número de rings antes del descuelgue, parámetros de la aplicación, nombre de la base

de datos para las consultas, y en parámetros en el flujo de programa para ir recuperando los datos de configuración de cada estado, por lo que el no tener un acceso a las bases de datos es un error principal que puede hasta no permitir que la aplicación se inicie, pero esto permite que la reconfiguración de la aplicación no necesite ni retocar código ni rescribir en muchos casos.

El siguiente diagrama Entidad-Relación muestra las tablas y las relaciones entre ellas.

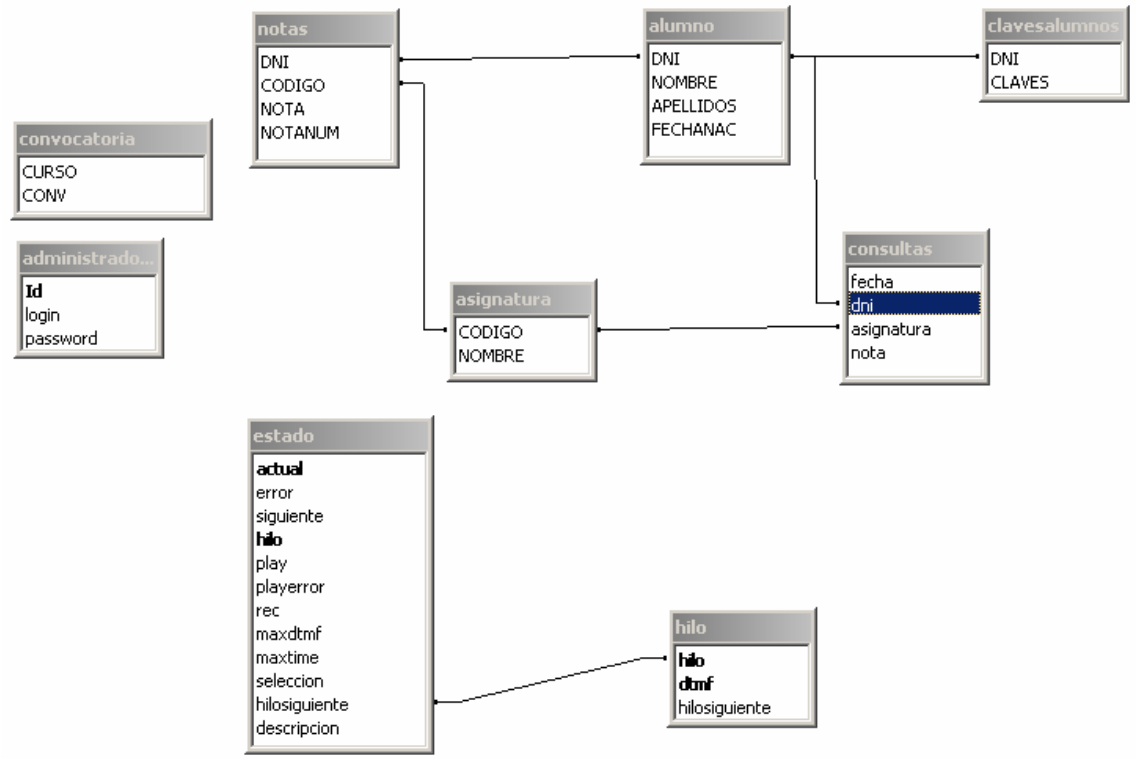

Figura 3.37: Diagrama Entidad - Relación

3.5.4 Mantenimiento de las bases de datos

Es este apartado se detallan las labores del administrador de sistema para el mantenimiento de las bases de datos de la aplicación.

Por un lado tenemos la base de datos de sistema en la que se almacenan los parámetros de configuración y los mensajes de la aplicación. En la tabla *config* se almacenan los datos de configuración, normalmente se modificaran al instalar la

aplicación por primera vez o en caso de algún cambio puntual como el número de rings, no tiene ningún tipo de mantenimiento especial. En la tabla *log* se almacenan los mensajes de la aplicación, esta tabla se consultara regularmente par comprobar que no hay mensajes de error en tiempo de ejecución y comprobar si la aplicación se ha iniciado correctamente, es interesante mantenerla con un cantidad de registros moderada para facilitar la búsqueda y comprobación de mensajes.

En la base de datos para las consultas si requiere un mayor mantenimiento ya que en esta se deben de actualizar los datos de las consultas. Por un lado podemos separar las tablas para la configuración del flujo de programa, para el histórico de consultas y para los datos de alumnos, asignatura y notas.

Para la configuración del sistema se utilizan las tablas *estado* e *hilo*, estas tablas se explicará su funcionamiento en el apéndice A, una vez que la aplicación queda configurada no requiere un mantenimiento a no se que se desee cambiar el flujo o el texto a reproducir.

Para el histórico de consultas se utiliza la tabla consultas, el único mantenimiento que requiere esta tabla es dejarla en blanco si en algún momento se quiere realizar una lectura de consultas para realizar alguna estadística o estudio, estos cambios se realizarán desde el menú administrador comentado en el capítulo siguiente.

Para los datos de las consultas se utilizan las tablas *Alumno*, *Asignatura*, *Notas,* y *Convocatoria*. Estas tablas interactúan directamente con Agora, la base de datos de Gestión Académica, por lo tanto su mantenimiento es aparte, para el administrador de esta aplicación estas tablas estarán siempre actualizados, luego no tendrá que preocuparse por ello.

3.6 Interfaz Administrador

En un principio sólo se dispondrá de un administrador que será el que lleve a cabo un seguimiento de la aplicación. El administrador podrá realizar las siguientes tareas:

- Historial de consultas
- Inicializar tabla de consultas

- Modificar password
- Restablecer clave de alumno

Siempre que el administrador quiera acceder a esta interfaz se le mostrará la siguiente pantalla, donde tendrá que autentificarse como administrador.

Se comprobará el login y el password en la tabla administrador de la base de datos de Upct como ya se comentó en el apartado anterior.

Figura 3.38: Acceso Administrador

Una vez que se haya autentificado como administrador se le mostrará el siguiente menú con las cinco opciones que puede realizar:

Figura 3.39: Menú principal del Administrador

Si pinchamos en historial de consultas se mostrará una pantalla como la siguiente:

Alumno	Código	Asignatura	Fecha	Nota	Calificación
77707133	122213017	TECNOLOGIA DE LA CONGELACION DE ALIMENTOS	Wed Jan 11 09:35:24 2006	9.00	Sobresaliente
Alumno	Código	Asignatura	Fecha	Nota	Calificación
77707133	122213017	TECNOLOGIA DE LA CONGELACION DE ALIMENTOS	Wed Jan 11 09:35:24 2006	9.00	Sobresaliente
Alumno	Código	Asignatura	Fecha	Nota	Calificación
77707133	122213017	TECNOLOGIA DE LA CONGELACION DE ALIMENTOS	Wed Jan 11 17:16:35 2006	9.00	Sobresaliente
Alumno	Código	Asignatura	Fecha	Nota	Calificación
77707133	122213017	TECNOLOGIA DE LA CONGELACION DE ALIMENTOS	Wed Jan 11 17:42:40 2006	9.00	Sobresaliente
Alumno	Código	Asignatura	Fecha	Nota	Calificación
77707133	122213017	TECNOLOGIA DE LA CONGELACION DE ALIMENTOS	Thu Jan 12 09:09:13 2006	9.00	Sobresaliente
Alumno	Código	Asignatura	Fecha	Nota	Calificación
77707133	122213017	TECNOLOGIA DE LA CONGELACION DE ALIMENTOS	Thu Jan 12 09:09:13 2006	9.00	Sobresaliente

Figura 3.40: Página Historial de consultas.

Donde aparecen en un principio el número total de consultas de cada mes seguido de una con un listado de las mismas. Además si pinchamos en el DNI de cualquier alumno de la lista obtenemos una pantalla con el nombre del alumno y el número total de consultas realizadas por este.

Figura 3.41: Seguimiento de un alumno particular

NOTA: El número de consultas no es total en toda la historia de la aplicación, sino que es el número de consultas realizadas desde la última vez que el administrador reinicializó la tabla de consultas.

La opción de inicializar tabla de consultas permite borrar por completo esta tabla, por este motivo, el administrador debe estar muy seguro de lo que hace, ya que se perderán todas las consultas realizadas hasta el momento:

Figura 3.42: Último paso para borrar el historial de consultas

Si de verdad quiere borrar los datos de las consultas recibirá el siguiente mensaje de confirmación:

Figura 3.43: Mensaje de confirmación de que se han borrado todas las consultas

Si el administrador desea cambiar su contraseña deberá rellenar el siguiente formulario:

Figura 3.44: Cambio de password de administrador

Puede ocurrir que un alumno haya olvidado su clave de acceso, es decir, que la modificara y no la recuerde. Para estos casos el administrador tiene la opción restablecer la clave de ese alumno, tan solo introduciendo el dni del mismo, la clave de acceso se restablecerá a la fecha de nacimiento como lo era inicialmente.

Figura 3.45: Restablecer clave de alumno

3.7 Aplicación

3.7.1 Introducción

En este apartado se describe la aplicación para capacitar al administrador del sistema a su instalación y mantenimiento e introducir los conceptos para la modificación y ampliación del código C en caso de que sea necesario o se desee añadir nuevas funcionalidades a la aplicación.

La aplicación ha sido escrita en su totalidad con lenguaje C, el API proporcionado por Dialogic incluye los ficheros de cabecera .h y las librerías estáticas y dinámicas .lib y .dll en lenguaje C, al igual que el API proporcionado por Verbio, la aplicación es estructurada en funciones y threads para que los canales trabajen concurrentemente.

3.7.2 Instalación y configuración

La aplicación esta compuesta por los siguientes directorios:

- Directorio raíz, es independiente de donde este, en el se encuentran todos los directorios de la configuración pro defecto.
- Directorio *db*, este contiene las bases de datos.
- Directorio *inc*, este contiene los ficheros de cabecera .h.
- Directorio *o*, este contiene los ficheros objeto para la compilación .o.
- Directorio *src*, este contiene los ficheros con el código fuente C de la aplicación .c.

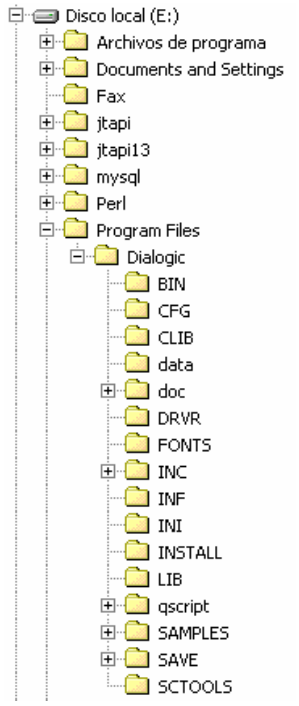

Figura 3.46: Directorios del *System Release 5.1.1*

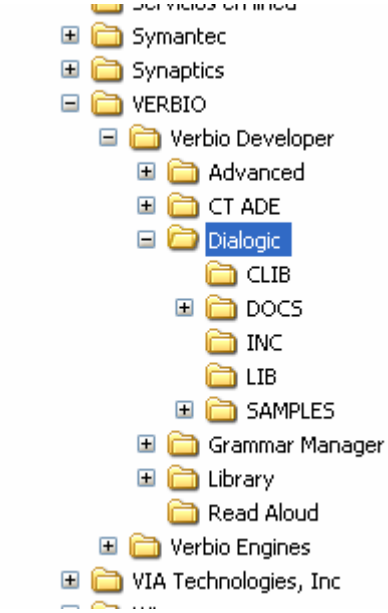

Figura 3.47: Directorios del *VerbioDeveloper*

Además son necesarios los ficheros .h y .lib de Dialogic para la compilación, estos se encuentran en el directorio donde se ha instalado el *System Release*, en nuestro caso *c:\Program files\Dialogic*, en los subdirectorios *inc* y *lib* respectivamente. También

son necesarios los fincheros .h y lib de Verbio para la compilación, estos se encuentran en el directorio que se ha istalado el VerbioDevolper, en nuestro caso c:\Archivos de programa\Verbio\VerbioDeveloper\Dialogic, en los subdirectorios inc\ y lib\.

Para instalar la aplicación solo son necesarios el directorios *db* y el fichero .exe de la aplicación, se copia todo esto a la ruta que se desee y la aplicación ya esta preparada para ejecutarse, previamente configuradas e instaladas las bases de datos.

1- Copiar la aplicación con sus directorios al lugar donde se desee.

Configuración del conector de las bases de datos con la herramienta del SO orígenes de datos (*panel de control-herramientas administrativas*).

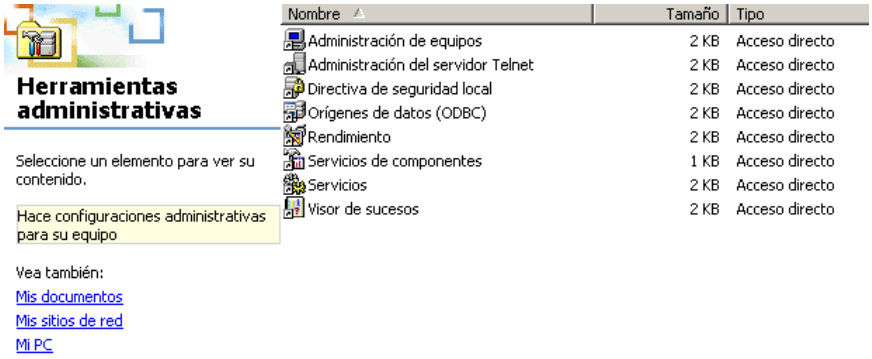

Figura 3.48: Herramienta orígenes de datos del panel de control

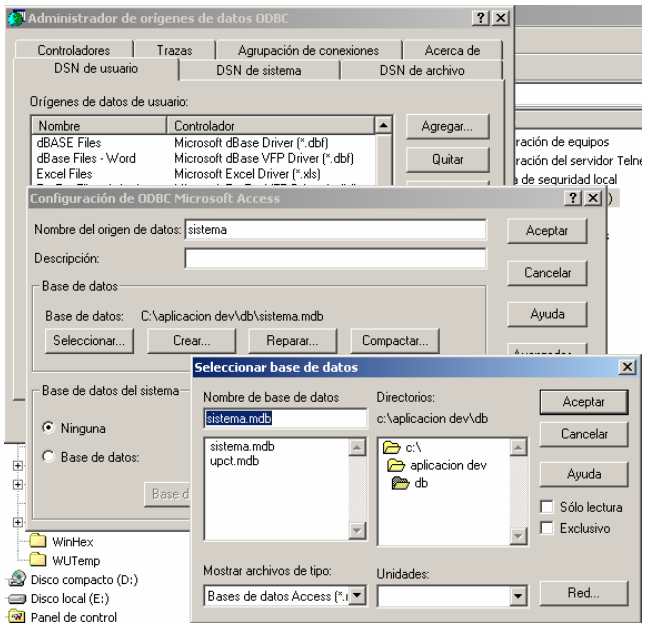

Figura 3.49: Creación del conector a la base de datos del sistema

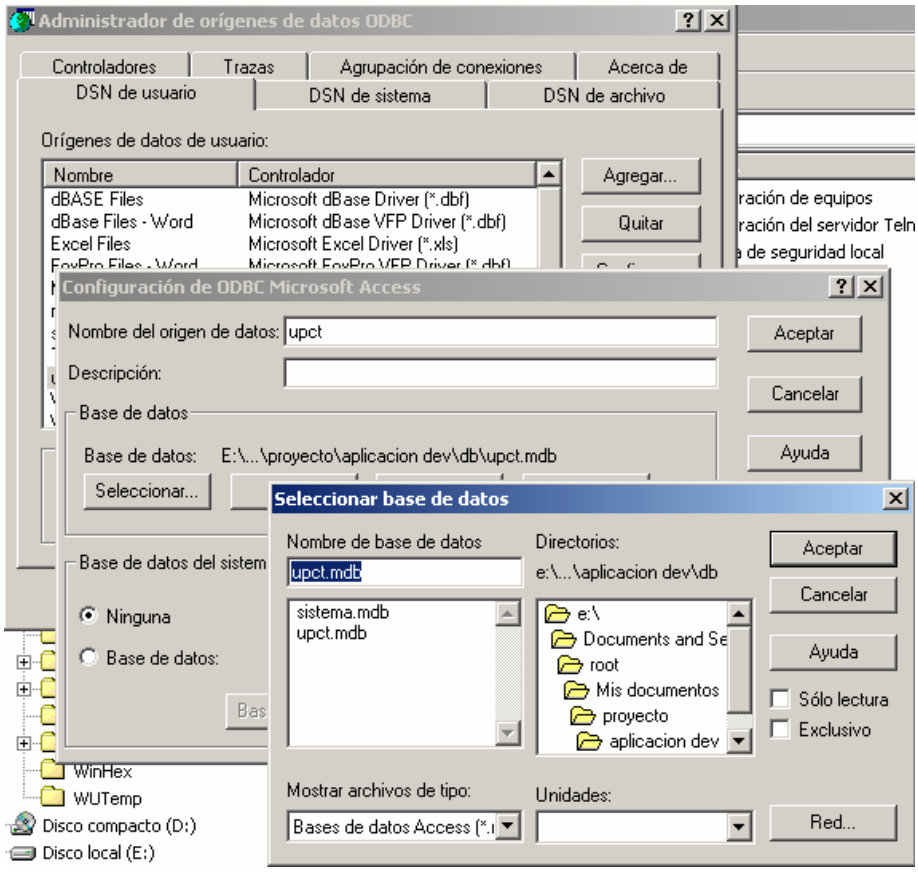

Figura 3.50: Creación del conector a la base de datos

2- Introducción de datos en las tablas para poder realizar consultas y reinicializar las tablas de *log* y *consultas* para que comiencen vacías.

3- Ejecutar la aplicación.

4- Comprobar el la tabla *log* que se ha inicializado sin errores.

3.7.3 Mantenimiento de la aplicación

El único mantenimiento que lleva la aplicación en situaciones de funcionamiento normal son los de las bases de datos (y de eso ya se encarga Gestión Académica), no obstante en el siguiente capítulo se explica la estructura y conceptos del código C para que pueda ser modificado o añadidas nuevas funciones.

Si hubiese alguna situación no prevista en tiempo de ejecución que generase un error grave y no permitiese que la aplicación fuese estable habría que modificar el código y recompilar.

Capítulo 4: Arquitectura de la Aplicación

4.1 Introducción

En este capítulo se resumirán los aspectos fundamentales de la implementación de la aplicación. El código fuente completo se encuentra en el directorio de la aplicación del CD del proyecto.

La aplicación ha sido desarrollada en su totalidad en lenguaje C, en dos ficheros pfc.c y pfc.h los cuales contienen el código de la aplicación y las definiciones, prototipos, etc respectivamente. Todas las configuraciones de parámetros son leídas de las tablas de las bases de datos permitiendo esto la modificación y configuración parcial sin necesidad de rescribir código ni recompilar.

Las funciones se pueden dividir en varios grupos dependiendo del tipo de acción que realicen, a continuación se muestran los prototipos de todas las funciones que se han implementado.

```
//Prototipos de funciones.

//Funciones Threads.
DWORD WINAPI DialogicThread(PVOID pvThreadParm);
DWORD WINAPI rutina(PVOID pvThreadParm);

//Funciones de inicio y fin.

int canalInit(int channum);
void canalQuit(int channum);

//Funciones utilidades.

int  set_hkstate( int channum, int state );
int  get_channum( int chtsdev );
int  get_digits( int channum);
int  play( int channum );
int  playError(int channum);
int  record( int channum );
```

```
void errorType(int channum,char *error);
void copyDB(int channum);
int  writeLog(char *s);
int construirFecha(int channum);
void setDate(int channum);
int  setParam(void);
int  formatNumber(char *numbers);
void channelState(int channum,char *state);
int  transfer(int channum,int ext);
int  transferOpe(int channum);

//Funciones Reset.

int resetChannel(int channum);
int resetState(int channum);

//Funciones GET.

int getAlumno(int channum);
int getSignature(int channum);
int getNote(int channum);
int getState(int channum);
int getOption(int channum);
int getSelOpeAuto(int channum);
int getClave(int channum);
int getConvocatoria(int channum);

//Funciones SQL.

int sqldisconn(DB *db);
int sqlconn(DB *db);
int sqlexec(DB *db);
int sqlSearchTableSignature(int channum);
int sqlSetConsulta(int channum);
int sqlSetConsulta2(int channum, int dni, int codigo, float nota);
int sqlGetParam(DB *db,unsigned char *s,int column);

//Funciones estados.

int stateProcess(int channum);
int setNextState(int channum);

int st_opcion(int channum);
int st_dni(int channum);
int st_passwd(int channum);
```

```
int st_signaturecod(int channum);
int st_signaturecod2(int channum);
int st_operator(int channum);
int st_goodBye(int channum);
int st_petpass1(int channum);
int st_petpass2(int channum);
int st_bienvenida(int channum);
int st_step(int channum);
int st_clave(int channum);
int st_conv(int channum);



//Funciones manejadoras de eventos.

long cst_hdlr(unsigned long x);
long callp_hdlr(unsigned long x);
long play_hdlr(unsigned long x);
long record_hdlr(unsigned long x);
long getdig_hdlr(unsigned long x);
long sethook_hdlr(unsigned long x);
long error_hdlr(unsigned long x);
long fallback_hdlr(unsigned long event_handle);

//Funciones de Verbio

DWORD WINAPI ThreadProc3(LPVOID);
long PlayStr(int channum, const char *str);

long GetDig(int dev, DV_DIGIT *digitp);
char *strcatdig(char *str, const char *dig);
void vx_error(const char *function, int chdev);
void dx_error(const char *function, int chdev);
void CheckWords(const char *vcb, int language);
int vx_getlicmode();
typedef int   (VERBIOAPI *ptr_vx_libinit)(int reserved);
typedef int   (VERBIOAPI *ptr_vx_prevcb)(const char *fileName, unsigned int mode);
typedef int   (VERBIOAPI *ptr_vx_setvcb)(int chdev, const char *fileName, unsigned int mode);
typedef int    (VERBIOAPI *ptr_vx_recstr)(int chdev, DV_TPT *tptp, VX_RSP *rspp, unsigned int
mode);
typedef int    (VERBIOAPI *ptr_vx_recind)(int chdev, int maxind, int *index, float *score, unsigned
int mode);
typedef char* (VERBIOAPI *ptr_vx_word)(int chdev, int ind);
typedef int    (VERBIOAPI *ptr_vx_playstr)(int chdev, const char *string, DV_TPT *tptp, unsigned
int mode);
```

```
typedef void  (VERBIOAPI *ptr_vx_clrrsp)(VX_RSP *rspp);
typedef long  (VERBIOAPI *ptr_ATVX_LASTERR)(int dev);
typedef char* (VERBIOAPI *ptr_ATVX_ERRMSGP)(int dev);
typedef int   (VERBIOAPI *ptr_vx_set_lasterr)(int chdev, long error);
typedef int   (VERBIOAPI *ptr_vx_nbest)(int chdev, int maxind, int *index, float *score, int ibest,
unsigned int mode);
typedef int   (VERBIOAPI *ptr_vx_getparm)(int chdev, unsigned long parm, void *valuep);
typedef int   (VERBIOAPI *ptr_vx_setparm)(int chdev, unsigned long parm, const void *valuep);
typedef int   (VERBIOAPI *ptr_ATVX_NIND)(int chdev);
typedef int   (VERBIOAPI *ptr_vx_termplaystr)(int chdev);
typedef int   (VERBIOAPI *ptr_vx_termrecstr)(int chdev);
typedef int   (VERBIOAPI *ptr_vx_chkwrd)(const char *word, int language);
typedef int   (VERBIOAPI *ptr_vx_prevcbex)(const char *fileName, unsigned int mode, int
*lpiword);
typedef int   (VERBIOAPI *ptr_vx_ApplyDictionary)(const char *inFileName, const char
*outFileName, const char *dicFileName);
typedef int   (VERBIOAPI *ptr_vx_setcd)(int chdev, unsigned int mode);
typedef char* (VERBIOAPI *ptr_vx_wordex)(int chdev, int ind, int pos);
typedef int   (VERBIOAPI *ptr_vx_loadvcb)(int chdev, const char *fileName, unsigned int mode);
typedef int   (VERBIOAPI *ptr_vx_loadcd)(int chdev, unsigned int mode);
typedef int   (VERBIOAPI *ptr_vx_activatevcb)(int chdev, int vcbhandle, unsigned int mode);
typedef int   (VERBIOAPI *ptr_vx_deactivatevcb)(int chdev, int vcbhandle, unsigned int mode);
typedef int   (VERBIOAPI *ptr_vx_unloadvcb)(int chdev, int vcbhandle, unsigned int mode);
typedef int   (VERBIOAPI *ptr_vx_recstr_open)(int chdev, VX_RSP *rspp, unsigned int mode);
typedef int   (VERBIOAPI *ptr_vx_recstr_write)(int chdev, const void* buffer, unsigned int n);
typedef int   (VERBIOAPI *ptr_vx_ttsSetDictionary)(int chdev, int mcLang, const char *filename);
typedef int   (VERBIOAPI *ptr_vx_ttsSetAbbreviations)(int chdev, int mcLang, const char
*filename);
typedef int   (VERBIOAPI *ptr_ATVX_IVCB)(int chdev);
typedef long  (VERBIOAPI *ptr_vx_GetDllVersion)(unsigned long *fileverp, unsigned long
*prodverp);
typedef int   (VERBIOAPI *ptr_vx_libclose)();
typedef int   (VERBIOAPI *ptr_vx_recstr_wait)(int chdev, int ms);
typedef int   (VERBIOAPI *ptr_vx_recstr_release)(int chdev);
typedef int   (VERBIOAPI *ptr_vx_playstr_wait)(int chdev, int ms);
typedef int   (VERBIOAPI *ptr_vx_playstr_release)(int chdev);
typedef int   (VERBIOAPI *ptr_ATVX_BUILTIN)(int chdev);
typedef int   (VERBIOAPI *ptr_vx_asr_init)(const char* configuration, const char* defasrlng);
typedef int   (VERBIOAPI *ptr_vx_tts_init)(const char* configuration, const char* defttslng);
typedef int   (VERBIOAPI *ptr_vx_getasrlic)(const char* configuration);
typedef int   (VERBIOAPI *ptr_vx_getttslic)(const char* language);
typedef int   (VERBIOAPI *ptr_vx_SetDictionary)(int chdev, const char *mcLang, const char
*filename);
typedef int   (VERBIOAPI *ptr_vx_SetAbbreviations)(int chdev, const char *mcLang, const char
*filename);
```

```
typedef void  (VERBIOAPI *ptr_vx_srvclose)(const char* server);
typedef int   (VERBIOAPI *ptr_vx_prevcbex2)(const char *fileName, unsigned int mode, int *lpiword, const char *language);
```

Tabla 4.1: Prototipos de las funciones de la aplicación

El funcionamiento básico de la aplicación es el siguiente, se define un objeto llamado estado el cual siempre actúa de la misma manera independientemente de la función que deba realizar, obtener un numero del usuario mediante la marcación de dígitos, realizar una consulta, etc, esto se realiza así para reutilizar el código y simplificar la programación.

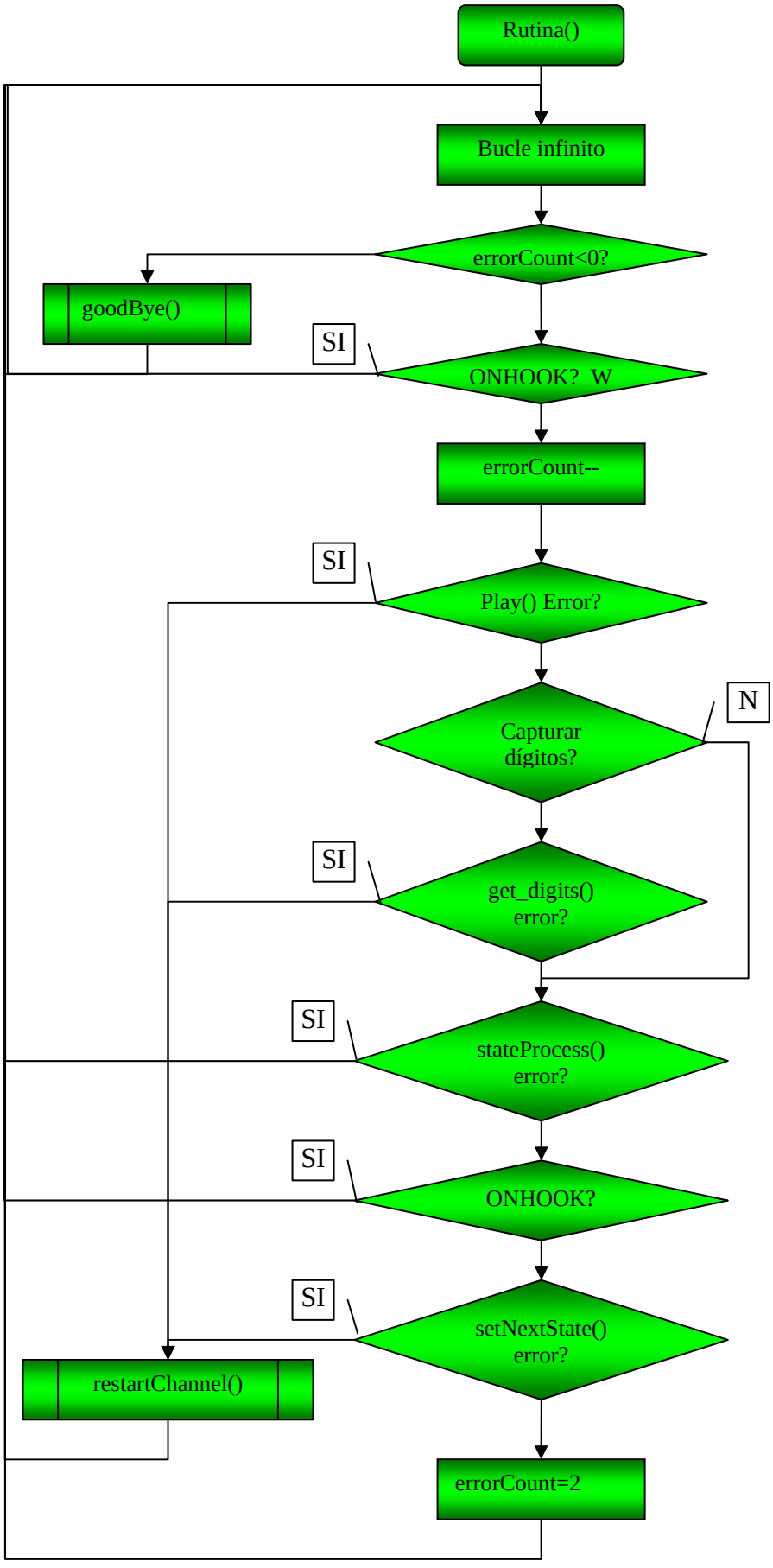

Figura 4.1: Diagrama de flujo de la función rutina()

En la tabla de estado de la base de datos UPCT se almacenan los parámetros de cada estado, numero de dígitos a capturar, tiempo para capturar dígitos, texto a reproducir, etc…, y *selección* que es numero que identifica la función que se realizara en la función stateProcess(). Este numero esta definido en el fichero upct.h, los definidos en la aplicación son:

```
//Definición de estados.
#define ST_OPCION       0      /* Menu de selección. */
#define ST_DNI          11     /* Introducción de DNI. */
#define ST_PASSWD       12     /* Introducción de contraseña. */
#define ST_SIGNATURECOD  13    /* Introducción de codigo de  asignatura. */
#define ST_SIGNATURECOD2 85    /* Lectura de notas de todas las asignatuas. */
#define ST_PETPASS1     15     /* Introducción de nueva contraseña. */
#define ST_PETPASS2     16     /* Introducción de nueva contraseña. */
#define ST_OPERATOR     21     /* Menu de operadora automatica. */
#define ST_BIENVENIDA   41     /* Reproducción de mensaje. */
#define ST_GOODBYE      91     /* Liberacion de canal. */
#define ST_STEP         99     /* Salto de hilo. */
#define ST_CLAVE        47     /*Comprobación de la clave. */
#define ST_CONV         57     /* Convocatoria actual*/
```

Tabla 4.2: Definición de las funciones de estados

Para programar un estado en la tabla de *estado* hay que poner el número correspondiente a la función que se realiza definida en la tabla anterior. En la función *stateProcess()* se ejecutara la función con el mismo nombre que la etiqueta pero en minúscula según se ve en el código de la función *stateProcess():*

```
int stateProcess(int channum){
  int rc;
switch(dxinfo[channum].state.seleccion){
    case ST_OPCION:
      rc=st_opcion(channum);
      break;
```

```
        case ST_CONV:
          rc=st_conv(channum);
          break;
        case ST_DNI:
          rc=st_dni(channum);
          break;
        case ST_PASSWD:
          rc=st_passwd(channum);
          break;
        case ST_CLAVE:
          rc=st_clave(channum);
          break;
        case ST_SIGNATURECOD:
          rc=st_signaturecod(channum);
          break;
        case ST_SIGNATURECOD2:
          rc=st_signaturecod2(channum);
          break;
        case ST_OPERATOR:
          rc=st_operator(channum);
          break;
        case ST_PETPASS1:
          rc=st_petpass1(channum);
          break;
        case ST_PETPASS2:
          rc=st_petpass2(channum);
          break;
         case ST_BIENVENIDA:
          rc=st_bienvenida(channum);
          break;
        case ST_GOODBYE:
          rc=st_goodBye(channum);
          break;
       case ST_STEP:
          rc=st_step(channum);
          break;
        default:
            resetState(channum);
      }//end switch
      if(rc<0)
        playError(channum);
      return rc;}
```

Tabla 4.3: función *stateProcess()*

Para añadir nuevas funcionalidades solo hay que añadir una etiqueta correspondiente, un nuevo case en la función *stateProcess*() y la función con el mismo nombre que la etiqueta en minúscula, luego en la tabla estado en el campo de selección habría que poner el numero de función que se ha definido en el fichero upct.h. Los argumentos de entrada y salida de las funciones van definidos en el fichero de cabecera upct.h.

Cada canal tiene un hilo, *Thread*, que se ejecuta concurrentemente para permitir la consulta simultanea de todos los canales, la escalabilidad de la aplicación es muy sencilla ya que con solo añadir nuevas tarjetas y modificar el número de canales que se quieren utilizar es suficiente por que se crearía un *thread* para cada canal.

A continuación mostramos el esquema de estados e hilos programados en nuestra aplicación:

estado : Tabla

actua	error	siguie	hilo	play	playerror	rec	maxdtm	maxtime	seleccio	hilosiguiente	descripcion
0	0	1	0	Bienvenido al Se			0	0	41	0	Mensaje de bienvenida
1	0	2	0	Porfavor, Introdu	No existe		8	15	11	0	Petición de DNI
2	0	3	0				0	0	47	0	Comprobación de que existe
3	0	4	0	Porfavor, introdu	Su clave		8	15	12	0	Petición de clave de acceso
4	0	5	0				0	0	57	0	Comprobación de clave correcta
5	0	6	0	Para consultar l	Esa entra		1	10	0	0	Menú de selección
5	0	40	91	Gracias por usa			0	0	91	9	Mensaje de despedida
6	0	7	1	Por favor, introdu	El código		9	15	13	1	Petición de código de asignatura
6	0	7	2				0	0	85	1	Lectura de notas todas las asignaturas
6	0	7	9	Introduzca la nu			8	15	15	9	Petición de nueva clave de acceso
6	0	40	91	Gracias por usa			0	0	91	0	Mensaje de despedida
7	0	5	1	Para realizar otr	Esa entra		1	5	0	1	Menú de selección
7	0	5	2	Para realizar otr	Esa entra		1	5	0	2	Menú de selección
7	0	9	9	Vuelva a introdu	No se pu		8	15	16	9	Petición de nueva clave por segunda vez
8	98	9	9				0	0	31	9	Cambio de clave de acceso
9	0	7	9				0	0	99	1	
0	0	0	0				0	0	0	0	

Tabla 4.4: Tabla estado de estados

hilo		
hilo	**dtmf**	**hilosiguiente**
0	1	1
0	2	2
0	3	9
0	4	91
1	1	0
1	2	91
2	1	0
2	2	91

Tabla 4.5: Tabla de hilo

La tabla hilo se utiliza para poder reutilizar los dígitos DTMF de un menú de selección a otro y poder identificar a que hilo salta el flujo de la aplicación.

Es muy importante, a la hora de compilar el proyecto, agregar las librerías necesarias para el linkado. Para ello lo que debemos hacer es lo siguiente:

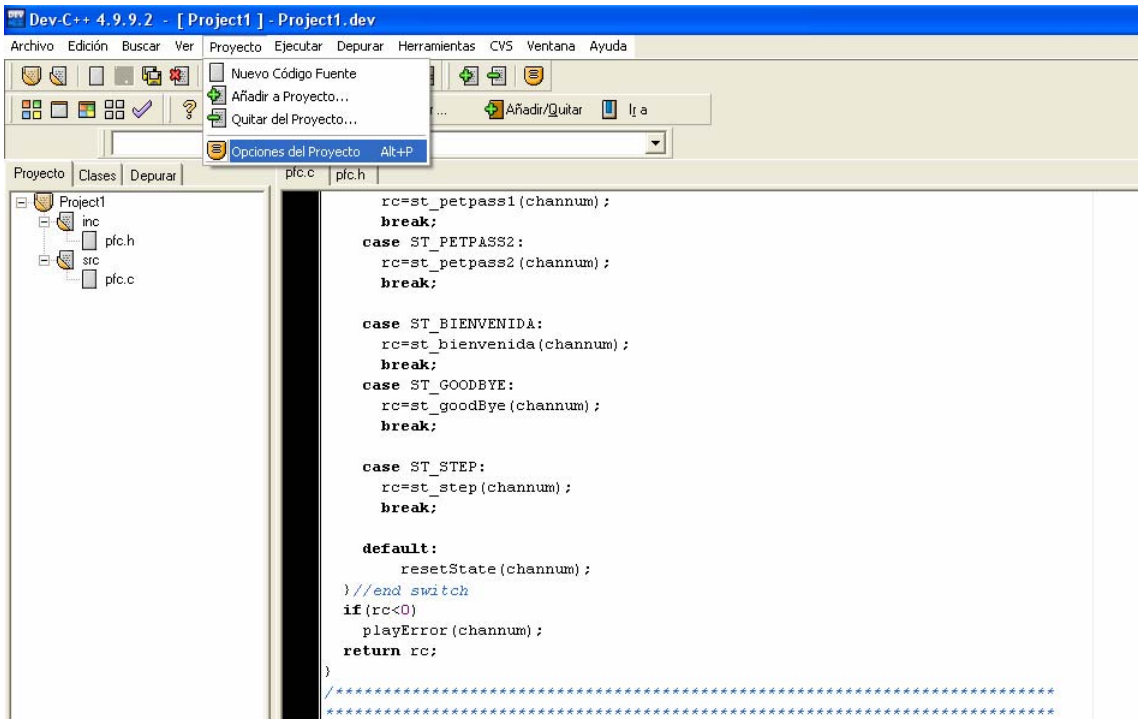

Figura 4.2: En proyecto, opciones de proyecto

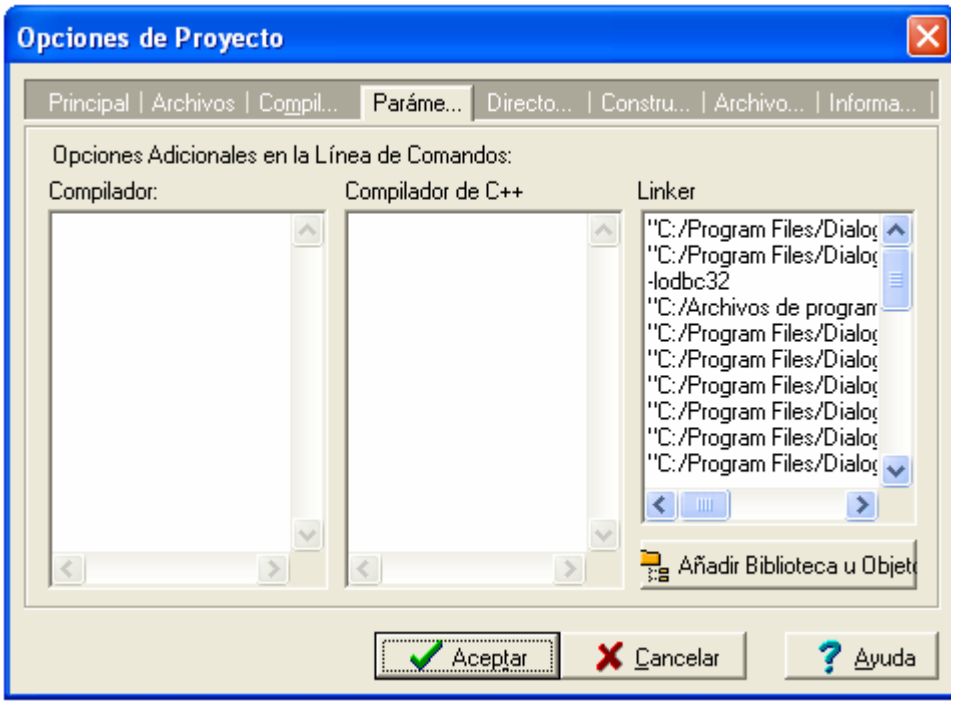

Figura 4.3: Menú de opciones de proyecto

```
"C:/Program Files/Dialogic/LIB/LIBDXXMT.lib"
"C:/Program Files/Dialogic/LIB/libsrlmt.lib"
-lodbc32
"C:/Archivos de programa/VERBIO/Verbio Developer/Dialogic/LIB/vxxxlib.lib"
"C:/Program Files/Dialogic/LIB/libgc.lib"
"C:/Program Files/Dialogic/LIB/Csapi32x.lib"
"C:/Program Files/Dialogic/LIB/Csgui.lib"
"C:/Program Files/Dialogic/LIB/DCMObjps.lib"
"C:/Program Files/Dialogic/LIB/dlreg.lib"
"C:/Program Files/Dialogic/LIB/kvscscu.lib"
"C:/Program Files/Dialogic/LIB/libd42mt.lib"
"C:/Program Files/Dialogic/LIB/libdtimt.lib"
"C:/Program Files/Dialogic/LIB/LIBDXXMT.lib"
"C:/Program Files/Dialogic/LIB/libecmt.lib"
"C:/Program Files/Dialogic/LIB/libfaxmt.lib"
"C:/Program Files/Dialogic/LIB/libgc.lib"
"C:/Program Files/Dialogic/LIB/libgcis.lib"
"C:/Program Files/Dialogic/LIB/Libgcs7.lib"
"C:/Program Files/Dialogic/LIB/libipm_nettsc.lib"
"C:/Program Files/Dialogic/LIB/libisdnr4.lib"
"C:/Program Files/Dialogic/LIB/libmsir4.lib"
"C:/Program Files/Dialogic/LIB/libprds.lib"
"C:/Program Files/Dialogic/LIB/libprtms.lib"
"C:/Program Files/Dialogic/LIB/librtfmt.lib"
"C:/Program Files/Dialogic/LIB/libsrlmt.lib"
"C:/Program Files/Dialogic/LIB/LMode.lib"
"C:/Program Files/Dialogic/LIB/mnti.lib"
"C:/Program Files/Dialogic/LIB/NCMapi.lib"
"C:/Program Files/Dialogic/LIB/sctools.lib"
"C:/Program Files/Dialogic/LIB/syntellect.lib"
"C:/Program Files/Dialogic/LIB/tascscu.lib"
```

Tabla 4.6: Librerías necesarias para linkar

Capítulo 5: Conclusiones y líneas futuras

En este Proyecto final de carrera se pueden contemplar las principales ventajas de los sistemas de telefonía interactiva IVR:

- **Reducción de costes:** los sistemas de atención telefónica automatizados requieren unos costes de mantenimiento notablemente menores a los proporcionados por operadores humanos.
- **Incremento de la calidad de servicio:** Un sistema automático permite dar servicio al cliente no sólo durante determinados periodos del día, sino las 24 horas del día y 365 días al año.
- **Recursos optimizados:** utilizar un sistema automatizado para realizar tareas que no requieran de personal especializado, permite distribuir de forma más optimizada los recursos humanos de la empresa asignándoles tareas más adecuadas a su perfil.

Además con la incorporación del sintetizador de voz todavía se ha optimizado más el servicio dado, ya que el sistema sobre el que se partió para realizar este proyecto final de carrera, constaba de unos mensajes de voz grabados que se iban reproduciendo según el flujo de la aplicación.

Nuestra aplicación, nos ha proporcionado una visón de lo que es implementar una aplicación compleja IVR y de cuales son los escenarios en los que este tipo de aplicaciones pueden ofrecer un servicio útil. Además nos ha abierto muchas nuevas puertas para la continuación del desarrollo de sistemas y herramientas de desarrollo IVR.

Con las conclusiones de este proyecto se pueden proponer varias líneas futuras para la continuación de esta línea de desarrollo que mejorarían aún más la accesibilidad y la utilidad de la aplicación, como pueden ser:

- Sistemas IVR en los que la interacción del usuario vaya un paso más allá utilizando sistemas de reconocimiento de voz para que el sistema sea más ameno y sencillo de utilizar o para ofrecer sistemas que ayuden a discapacitados. La casa Verbio proporciona también herramientas de

reconocimiento de voz, y como ya tenemos instalado todo el software sólo necesitaríamos comprar las licencias.

- Proporcionar más funcionalidades útiles para el alumno, cómo podrían ser la fecha y hora de revisión del examen de la asignatura consultada, aunque para ello debería modificarse también la base de datos Agora de Gestión Académica.
- El desarrollo de clase para permitir un desarrollo más rápido de ampliaciones utilizando los fundamentos de la programación orientada a objetos (POO).

En definitiva, los sistemas IVR nos abren un abanico de posibilidades a la hora realizar sistemas de consultas y gestión de llamadas de forma fácil y rápida. Además, hoy en día con la integración de la telefonía y los computadores prácticamente todas las tareas que realiza un operador a través de un servicio telefónico pueden ser automatizadas hasta tal punto que una aplicación IVR ate todos los cabos puede optimizar las tareas reduciendo costes y tiempos.

Apéndice A: Funcionamiento de un conversor Texto a Voz

Una conversión automática de texto a voz supone una serie de procesos de tipo muy diverso. Por un lado, es necesario realizar un conjunto de tareas de tipo lingüístico que, partiendo del análisis del texto de entrada, puedan proporcionar datos útiles para la correcta y natural lectura del texto. Por otro lado, es necesario generar una señal de voz mediante métodos electrónicos, lo cual supone el empleo de técnicas de procesado digital de voz.

Un Conversor Texto-Voz general puede ser representado mediante un diagrama de bloques como el siguiente, en el que se puede apreciar esta dualidad de tareas que es necesario llevar a cabo. Por un lado, aparecería un *bloque de procesado lingüístico* y por otro lado un *bloque de síntesis de voz*. Dentro del primero se pueden distinguir los siguientes módulos fundamentales —aunque pueden existir algunos más—: preprocesador, categorizador, estructurador, conversor grafema-alófono, pausador y síntesis prosódica.

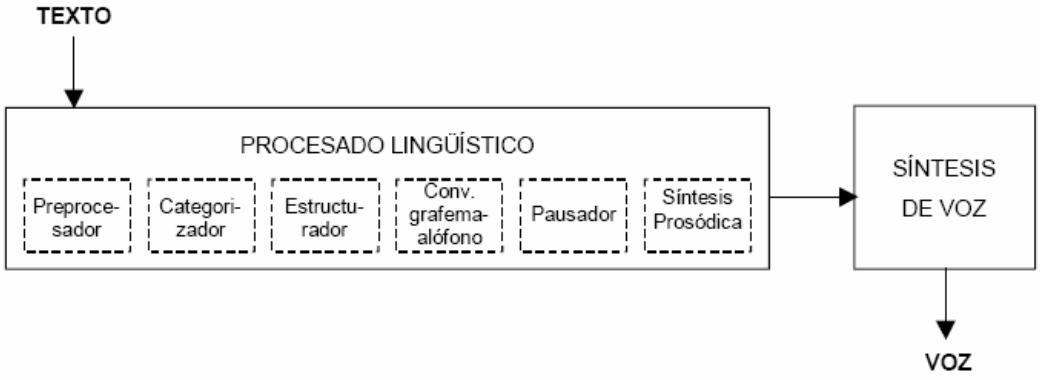

Figura A.1: Diagrama de bloques de un conversor texto a voz

La principal función del *bloque de procesado lingüístico*, es la de determinar la secuencia de sonidos que hay que producir para sintetizar correctamente el texto de entrada. Los resultados generados en este bloque se pasarán al bloque de síntesis, que procesará los datos y generará los sonidos. En este bloque se llevan a cabo

fundamentalmente dos tareas: obtener la cadena de sonidos (alófonos) correspondientes al texto de entrada, y obtener información prosódica para su producción.

La principal función del *bloque de procesado lingüístico*, es la de determinar la secuencia de sonidos que hay que producir para sintetizar correctamente el texto de entrada. Los resultados generados en este bloque se pasarán al bloque de síntesis, que procesará los datos y generará los sonidos. En este bloque se llevan a cabo fundamentalmente dos tareas: obtener la cadena de sonidos (alófonos) correspondientes al texto de entrada, y obtener información prosódica para su producción.

Una descripción breve de la funcionalidad de cada uno de estos bloques podría ser la siguiente:

- Preprocesador. Un texto puede contener expresiones que no están constituidas directamente por letras y palabras, y que es necesario interpretar para que puedan ser tratadas por el resto de los módulos que constituyen el bloque de procesado lingüístico del Conversor. El preprocesador se encarga de sustituir estas expresiones por la secuencia de caracteres alfabéticos que equivaldrían a la lectura que haría una persona.
- Categorizador. La principal tarea de este módulo es la de asignar categorías a las palabras con el objetivo de que el Conversor haga una lectura “con sentido” de los textos (añadiendo, por ejemplo, pausas que no venían marcadas ortográficamente). Las categorías que se asignan no sonexactamente categorías gramaticales, sino un conjunto de códigos que en muchos casos se corresponden con verdaderas categorías gramaticales, pues descubrir la estructura sintáctica sigue siendo una labor demasiado compleja si no se ponen restricciones a la gramática.
- Estructurador. La misión de este módulo es la de realizar un análisis sintáctico de las frases.
- Conversor grafema-alófono. Este módulo es el que se encarga de determinar cuál es la secuencia de alófonos que corresponde a la secuencia de letras de una frase. Hay que tener en cuenta que la correspondencia entre letras (grafemas) y

sonidos (alófonos) no es tal que siempre la misma letra produzca el mismo alófono, sino que la conversión depende de una serie de reglas.

- Pausador. El pausador se encarga de introducir, en el discurso, pausas adicionales a las que van indicadas mediante signos ortográficos. La principal razón para introducir pausas es simular el comportamiento real de un lector humano. La introducción de las pausas debe hacerse de acuerdo con la estructura sintáctica del mensaje para intentar reforzarla, ayudando así a la comprensibilidad del mismo.
- Síntesis prosódica. Los principales aspectos de la prosodia que se fijan en este módulo son, además de las pausas, la *duración* de los alófonos, la evolución temporal de la *frecuencia fundamental* (o contorno de *F*0) y el contorno de la *energía* o amplitud de la señal de voz.

La misión del *bloque de síntesis de voz* es la de generar sonidos tan similares a la voz como sea posible, presentando un alto grado de flexibilidad en cuanto a su capacidad para ser controlado, de modo que se pueda variar la realización de los sonidos. La información de entrada a este bloque incluye la secuencia de alófonos que hay que generar, y los datos de prosodia (típicamente, duración de los alófonos, contorno de frecuencia fundamental, y contorno de energía o amplitud).

Hay dos enfoques que, en cierto modo, determinan el tipo de sintetizador que se emplee. El primero de ellos, que podríamos denominar «modelo de sistema», intenta modelar, con mayor o menor detalle, el mecanismo de producción de la voz. Este enfoque ha dado origen a dos tipos de sintetizadores: los *sintetizadores articulatorios* y los *sintetizadores de formantes*. El segundo enfoque, que podríamos llamar «modelo de señal», es el que intenta modelar no el mecanismo de producción, sino la señal de voz; dentro de este enfoque se encuentran los *sintetizadores por concatenación*.

Apéndice B : API de Dialogic

En este apéndice se describe el paquete que proporciona Dialogic llamado *System Release*, la versión actual es la 6.0 y existe para los sistemas operativos Windows, en sus versiones NT, 2000 y XP, y Linux, en la distribución Red Hat 7.3, la versión anterior a esta es la 5.1.1 con *Service Pack 1*, esta ultima es la elegida para el desarrollo del proyecto por ser gratuita y por tener características muy equivalentes y mas que suficientes para el propósito del proyecto. Este paquete proporciona:

- Drivers.
- Firmware.
- Archivos de configuración.
- Entorno de desarrollo SDK.
- Programas de ejemplo.
- Proveedor de servicio TAPI.
- Documentación.
- API.
- Y software para las diferentes familias de tarjetas Dialogic (BRI, ISDN, antares, DM3, CSP, etc.).

En el capitulo 3 se describió la instalación y configuración de él, ahora comentaremos la documentación que proporciona acerca de la tarjeta D4/PCI usada en este proyecto.

Esta documentación se proporciona en varios documentos PDF, estos son:

- *1828-02.pdf*. Describe el contenido del Service Pack 5.1.1 para Windows.
- *dnacfgp.pdf*. Guía de configuración e instalación.
- *featgd3.pdf*. Guía de características.
- *icdcust.pdf*. Personalización de las herramientas de configuración
- *Installation Guide.pdf*. Guía de instalación.

- *pgmgd3.pdf*. Guía del programador de Windows.
- *pt_history.pdf*. Lista histórica de puntos y problemas resueltos.
- *release_guide.pdf*. Guía del System Release 5.1.1 de Windows.
- *release_update.pdf*. Actualización del System Release 5.1.1 de Windows.
- *srlgd3.pdf*. Guía de la librería estándar de *runtime* de Windows.
- *userguide.pdf*. Guía del usuario de Windows.

Solo haremos una descripción de los documentos *pgmgd3.pdf* y *srlgd3.pdf* que son los que nos proporcionan la información acerca del API para la programación de aplicaciones.

El *pgmgd3.pdf* contiene información para el programador describiendo el software con sus librerías y drivers, técnicas de programación, estructuras, parámetros y funciones, además incluye documentación de *Syntellect License Automated Attendant* que es una licencia para programación rápida y sencilla de operadoras automáticas. El índice abreviado de este documento es:

1. *Overview of Voice Documentation*
   - *1.1. How This Guide is Organized*
   - *1.2. Related Dialogic Publications*
2. *Overview of the Voice Software*
   - *2.1. Voice Driver*
   - *2.2. Voice Libraries*
   - *2.3. Voice Functions*
3. *Using the Voice Software*
   - *3.1. Voice Programming Requirements*
   - *3.2. Voice Programming Conventions*
4. *Syntellect License Automated Attendant*
   - *4.1. Overview of Automated Attendant Function*
   - *4.2. Syntellect License Automated Attendant Functions*
   - *4.3. How To Use the Automated Attendant Function Call*
5. *Voice Data Structures*
6. *Voice Device Parameters*
7. *Voice Function Reference*

Appendix A - Standard Run-time Library: Voice Device Entries and Returns

Appendix B - Voice Error Defines

Appendix C - DTMF and MF Tone Specifications

El srl*gd3.pdf* contiene información sobre la librería estándar de tiempo de ejecución describiendo los modelos de programación del API; sincronía, asíncrona y asíncrona extendida, además proporciona la documentación sobre el manejo de eventos. El índice abreviado de este documento es:

1. *SRL Overview*
2. *Basic SRL Programming Concepts for Dialogic Products*
3. *Basic SRL Programming Models*
4. *Advanced SRL Programming Concepts for Dialogic Products*
5. *Advanced SRL Programming Models*
6. *SRL Event Management Functions*
7. *SRL Standard Attribute Functions.105*
8. *DV_TPT Termination Parameter Table Structure .117*

Apéndice C: API de VERBIO

En este apéndice se describen los componentes que se han instalado a la hora de ejecutar el VerbioEngine.exe y el VerbioDeveloper.exe, la versión que se instaló es la 7.13 aunque hace un mes la casa Verbio ya dispuso la 7.14.

VerbioEngine.exe

Este componente sólo instala los elementos básicos para permitir el uso posterior de los módulos de reconocimiento y/o síntesis deseados. Una vez instalados los componentes específicos de Verbio, éste instalador comprueba que estén presentes los driver de las llaves de lincenciamiento, y si no están da la opción de instalarlos en ese momento.

VerbioDeveloper.exe

Dentro de los módulos adicionales para el cliente, encapsulados dentro de este instalador se encuentran los siguientes:

- Clientes disponibles
  - Verbio Dialogic Client
  - Verbio Library Client
  - Verbio Advanced Client
  - Verbio CT ADE Client
- Herramientas de Test y desarrollo
  - Verbio Grammar Manager
  - Verbio Read Aloud

De los cuatro clientes que ofrece Verbio, nosotros hemos usado el que más se amolda a nuestras necesidades: *Verbio Dialogic Client*, del cual tenemos:

- Software Devoloper Kit (SDK): conjunto de ficheros destinados a progrmar nuevas aplicaciones que incorporen reconocimiento y/o síntesis del habla (librerías, header, etc):

- Ejemplos: códigos de ejemplos de utilización del SDK
- Documentación: Descripción de las funciones contenidas en el SDK (Function Referente).

El Verbio Software Referente (Referencia de las funciones del Dialogic SDK), está incorporado en el CD del proyecto (**Dialogic-sdk_es.pdf**). El objetivo de este documento es describir el conjunto de funciones que constituyen el Dialogic SDK proporcionado en *Verbio*. Este SDK está diseñado pensando en aquellos integradores que trabajan en entornos de programación C/C++ sobre tarjetas Intel Dialogic, como es nuestro caso. Este documento se compone de:

- Capítulo 1: Introducción
- Capítulo 2: Descripción de las funciones, se describen todas las funciones del SDK
- Capítulo 3: Estructuras de datos y parámetros, se describen las estructuras y parámetros comunes a las funciones del SDK
- Capítulo 4: Códigos de ejemplo C/C++, se incluyen tres ejemplos de utilización de este SDK
- Apéndice A: Ficheros utilizados de ejemplo.

Como ya se ha comentado antes, toda esta documentación se encuentra en el CD del proyecto.

BIBLIOGRAFÍA

[1] Página Web del fabricante Ericsson

http://www.ericsson.com/enterprise/products/md110.shtml

[2] Dialogic: Intel Telecom. Products – Media Processing Boards

http://www.intel.com/design/network/products/telecom/boards/mediaprocessing.htm

[3] Página Web del fabricante Selta

http://www.selta.es

[4] Página principal de Verbio

http://www.verbio.com

http://www.verbio.com/reference/pdf/guide_es.pdf

[5] Guía de Library SDK

http://www.verbio.com/reference/pdf/library_sdk_es.pdf

[6] Guía de Advanced SDK

http://www.verbio.com/reference/pdf/advanced_sdk_es.pdf

[7] Intel® Dialogic® D/4PCI Datasheet
http://www.intel.com/network/csp/pdf/5795ds.pdf

[8] System Release 5.1.1 Feature Pack 1 for Windows

http://www.intel.com/network/csp/products/8506web.htm

[9] Bloodshed Dev-C++ resource site

http://www.bloodshed.net/devcpp.html

[10] Relación de las CTR en vigor Publicadas en el diario oficial de las comunidades europeas

http://www.setsi.mcyt.es/legisla/regtec.htm

[11]Telecom Board System Release Software

http://www.intel.com/network/csp/products/indx_aet.htm

[12] Sitio de descargas de Verbio

http://www.verbio.com/soportecnic.php?tema=downl#ASR%20&%20TTS%20Server

[13] Documentación Software Development Kit (SDK)

http://www.verbio.com/reference/pdf/dialogic_sdk_es.pdf